P. 1877/80

1 1990

informatyka

Sortowanie wg polskiego alfabetu
Język Lotos
Spis treści rocznika 1989

WYDAWNICTWO NOT SIGMA

Cena: 5800 zł

informatyka

Miesięcznik

ISSN 0542-9951. INDEKS 36124

nr 1 1990
Styczeń
Rok wydania XXV

W numerze:

Strona

Sprzęg graficzny jako system ekspertowy
Marek Hołyński 2

Porządkowanie zbiorów danych według polskiego alfabetu
Andrzej Dydek 6

Problemy implementacji języka Ada
Andrzej Paprocki 9

Okres istnienia oprogramowania (1)
Janusz Zalewski 12

Język Lotos (1). Specyfikacje typów danych
Zbigniew Huzar, Ludwik Kuźniarz 19

Uruchamianie urządzeń mikroprogramowanych (2)
Marek Gondzio, Marek Pawłowski, Jacek Wytrębowicz 24

Z KRAJU 27

Współczesne obszary zastosowań informatyki
Infosystem '89

TERMINOLOGIA III okł.

Polskie znaki i klawiatury w normach międzynarodowych

W najbliższych numerach:

- Richard C. Holt opisuje język programowania Turing, opracowany i stosowany na Uniwersytecie w Toronto.
- Lech Banachowski, Jarosław Deminet i Marek J. Lao podają propozycję nowego programu uniwersyteckich studiów informatycznych.
- Jan Grzegorz Cabański i in. charakteryzują automatyczny sterownik transmisji szeregowych dla PC XT/AT pod nazwą ASTER.
- Katarzyna Lubińska omawia zalety i rolę Ady w wytwarzaniu oprogramowania.
- Henryka Sobol charakteryzuje rozwiązanie programowego systemu sterowania bazą wideodanych oraz systemów przetwarzania obrazów.
- Mariusz Ormiński zaczyna cykl artykułów nt. lokalnej sieci komputerowej Ethernet.
- Mieczysław Kłopotek i in. opisują system pozyskiwania i weryfikacji wiedzy medycznej oznaczony symbolem SYS8688.

KOLEGIUM REDAKCYJNE:

Mgr Jarosław DEMINET,
dr inż. Wacław ISZKOWSKI,
mgr Teresa JABŁOŃSKA
(sekretarz redakcji),
Władysław KLEPACZ
(redaktor naczelny),
dr inż. Marek MACHURA,
dr inż. Wiktor RZECZKOWSKI,
mgr inż. Jan RYŻKO,
mgr Hanna WŁODARSKA,
dr inż. Janusz ZALEWSKI,
(zastępca redaktora naczelnego).

PRZEWODNICZĄCY
RADY PROGRAMOWEJ:

Prof. dr hab.
Juliusz Lech KULIKOWSKI

Materiałów nie zamówionych redakcja nie zwraca

Redakcja: 01-552 Warszawa,
Pl. Inwalidów 10 p. 128, 136
tel. 39-14-34

Wydawca: Wydawnictwo Czasopism i Książek Technicznych SIGMA NOT
Spółka z o.o.
ul. Biała 4,
00-950 WARSZAWA
skrytka pocztowa 1004

Druk:
PRASOWE ZAKŁADY GRAFICZNE
RSW „PRASA-KSIĄŻKA-RUCH"
ul. Dworcowa 13,
85-950 BYDGOSZCZ
Zam. 4368/89 E-3
Obj. 4,0 ark. druk. Nakład 6000 egz.

Cena egzemplarza 5800 zł

W sprawach ogłoszeń prosimy zwracać się bezpośrednio do Redakcji

WARUNKI PRENUMERATY

Prenumeratorzy zbiorowi – jednostki gospodarki uspołecznionej, instytucje i organizacje społeczne zamawiają prenumeratę dokonując wpłat wyłącznie na blankiecie „wpłata zamówienie" (jest to „polecenie przelewu" rozszerzone dla potrzeb Wydawnictwa o część dotyczącą zamówienia). Blankiety te będą dostarczane dotychczasowym prenumeratorom przez Zakład Kolportażu. Nowi prenumeratorzy otrzymają je po zgłoszeniu zapotrzebowania (pisemne lub telefoniczne) w Zakładzie Kolportażu, w Radach Wojewódzkich NOT bądź w Redakcjach czasopism.

Prenumeratorzy indywidualni – osoby fizyczne zamawiają prenumeratę dokonując wpłaty w UPT lub NBP na blankiecie NBP. Na odwrocie wszystkich odcinków blankietu należy wpisać tytuł czasopisma, okres prenumeraty, liczbę zamawianych egzemplarzy oraz wartość wpłaty. Wpłacać należy na konto: **Państwowy Bank Kredytowy III/O Warszawa nr 370015-7490-139-11.**

Prenumerata ulgowa – przysługuje wyłącznie osobom fizycznym – członkom SNT, studentom i uczniom szkół zawodowych. Warunkiem prenumeraty ulgowej jest poświadczenie blankietu wpłaty (przed jej dokonaniem) na wszystkich odcinkach pieczęcią Koła SNT, wyższej uczelni lub szkoły. Sposób zamawiania prenumeraty ulgowej jest taki sam jak prenumeraty indywidualnej. W prenumeracie ulgowej można zamówić tylko po jednym egzemplarzu każdego czasopisma.

Uwaga! Miesięcznik „Aura" może być zamawiany w prenumeracie ulgowej również przez uczniów szkół ogólnokształcących.

Prenumerata ze zleceniem wysyłki za granicę – zamawia się tak, jak prenumeratę indywidualną. Dodatkowo należy podać na blankiecie wpłaty nazwisko i dokładny adres odbiorcy.

Cena prenumeraty ze zleceniem wysyłki za granicę jest dwukrotnie wyższa.

Wpłaty na prenumeratę przyjmowane są w terminach:

- do 10 listopada na każdy kwartał, I i II półrocze oraz cały rok następny,
- do 28 lutego na II, III i IV kwartał oraz II półrocze,
- do 31 maja na III i IV kwartał oraz II półrocze,
- do 31 sierpnia na IV kwartał.

Zmiany w prenumeracie można zgłaszać pisemnie tylko w wyżej wymienionych terminach.

Informacji o prenumeracie udziela – Zakład Kolportażu Wydawnictwa NOT SIGMA (ul. Bartycka 20, 00-716 Warszawa), skr. poczt. 1004, 00-950 Warszawa, tel. 40-00-21 wew. 248, 249, 293, 297, 299 lub 40-30-86 i 40-35-89.

Egzemplarze archiwalne czasopism – można nabyć za gotówkę w Klubie Prasy Technicznej, Warszawa ul. Mazowiecka 12 (tel. 26-80-16), lub zamówić pisemnie. Zamówienia na egzemplarze archiwalne czasopism przyjmuje: Zakład Kolportażu, Dział Handlowy, 00-950 Warszawa, skr. poczt. 1004 (tel. 40-37-31), na rachunek dla instytucji lub za zaliczeniem pocztowym dla osób fizycznych.

Cena egzemplarza: normalna 5800 zł, ulgowa 1160 zł.

Uwaga! Podane ceny mają charakter wstępny i mogą ulec zmianie, w związku z powyższym Wydawnictwo zastrzega sobie prawo żądania dopłat do już opłaconej prenumeraty.

P. 1877/90

Wchodząc w ostatnie dziesięciolecie XX wieku należy stwierdzić, że dekada lat osiemdziesiątych dla INFORMATYKI nie była zbyt sprzyjająca. Wprawdzie zdołaliśmy odbudować załamaną po roku 1982 wielkość nakładu, osiągając w ub. r. poziom 8500 egzemplarzy, co w kategorii czasopism naukowo-technicznych jest już liczbą znaczącą, ale w dziesięcioleciu tym nastąpiły dwa, całkowicie niezależne od redakcji, znaczne załamania cyklu wydawniczego: pierwsze w 1982 r., gdy po wprowadzeniu stanu wojennego zawieszono okresowo działalność wydawniczą (zdołaliśmy wydać tylko sześć zeszytów), drugie w roku ubiegłym, gdy wskutek braku papieru musieliśmy zakończyć rok faktycznie na numerze sierpniowym. Szczególnie dotkliwie odczuliśmy załamanie ubiegłoroczne, ponieważ nastąpiło ono w momencie, gdy dzięki zmianie drukarni na Prasowe Zakłady Graficzne w Bydgoszczy, udało się nadrobić przeszło półroczne opóźnienie cyklu wydawniczego jakie wynikło w 1988 r. wskutek niesprawności dotychczasowej warszawskiej drukarni.

Intensyfikacja procesu uzdrawiania gospodarki powoduje, że w bieżącym roku INFORMATYKA, podobnie jak większość krajowych czasopism, a właściwie cały ruch wydawniczy w Polsce, zostanie poddana trudnemu egzaminowi. Egzaminowi, który powinien wykazać jej zdolność do funkcjonowania w mechanizmach ekonomicznych gospodarki rynkowej. W wyniku urealnienia rachunku ekonomicznego dwóch podstawowych składników kosztów działalności wydawniczej, jakimi są papier i usługi poligraficzne, cena naszego czasopisma w stosunku do ubiegłego roku musi wzrosnąć dwudziestokrotnie. Przyczyną tak znacznego skoku jest nie tylko fakt, że poprzednia cena została ustalona w trzecim kwartale 1988 r., ale przede wszystkim to, że w warunkach ostrego deficytu papieru od bieżącego roku zlikwidowano jego dotychczasową reglamentację. Doprowadziło to do ustalenia się na rynku krajowym cen tego podstawowego surowca produkcji wydawniczej na poziomie zbliżonym do cen światowych (ok. 1000 dolarów za tonę). Odczuliśmy to wszyscy już pod koniec ubiegłego roku na przykładzie galopujących cen prasy codziennej.

Tak znaczny wzrost cen papieru przy jednoczesnym odrzuceniu mechanizmów subwencjonowania z pewnością będzie w tym roku głównym czynnikiem rynkowej selekcji wszystkich wydawnictw krajowych, zwłaszcza periodyków. Nastąpi niewątpliwie samolikwidacja nie tylko tych tytułów, na które zapotrzebowanie społeczne jest niewielkie (były utrzymywane głównie dzięki dotacjom), ale również tych, które nie zdołają zrównoważyć wysokiej ceny treścią interesującą dostatecznie duży krąg odbiorców. Wydaje się, że INFORMATYKA jako jedyny w swej dziedzinie krajowy periodyk naukowo-techniczny ma większe niż czasopisma innych branż szanse nie tylko przetrwania, ale również dalszego rozwoju. Twierdzenie takie uzasadnia szybka, jakby na przekór głębokiemu kryzysowi gospodarczemu, informatyzacja wszystkich dziedzin naszego życia społecznego i gospodarczego.

Rozwój czasopisma, to w pierwszym rzędzie rozszerzenie kręgu jego odbiorców, a więc wzrost nakładu. Większy nakład, to z kolei obniżka kosztów jednostkowych czasopisma, stwarzająca warunki albo do obniżki jego ceny, albo też zwiększenia objętości i poprawy szaty graficznej. Obie te możliwości powiększają atrakcyjność czasopisma, a więc prowadzą do pozyskania nowych odbiorców i tym samym do dalszego wzrostu nakładu. Wzrost nakładu i atrakcyjności czasopisma pobudza również napływ ogłoszeń, które w istotny sposób wpływają na poprawę rentowności procesu wydawniczego, a tym samym tworzą przesłanki do dalszej obniżki ceny lub zwiększenia objętości.

Wracając do istoty podjętego tematu utrzymania dotychczasowej pozycji oraz rozwoju INFORMATYKI w twardych rygorach mechanizmów rynkowych należy stwierdzić, że warunkiem prowadzącym do tego celu jest takie zmodyfikowanie jej treści, aby w odczuwalny sposób wzrósł w niej udział informacji użytecznych dla maksymalnie szerokiego kręgu odbiorców. Krąg ten tworzą w przeważającej mierze osoby już działające lub zamierzające działać w obszarze zastosowań. Światowe trendy rozwoju informatyki wskazują coraz wyraźniej, że w środowisku związanym z tą dziedziną coraz większy procentowy udział mają właśnie użytkownicy. Potwierdzają to olbrzymie, sukcesywnie wzrastające nakłady tych czasopism, które są adresowane głównie do użytkowników.

W treści INFORMATYKI na przestrzeni ostatnich lat punkt ciężkości przesuwał się coraz bardziej na obszary zainteresowań środowiska naukowego. Wywoływało to wzrost krytycznych uwag znacznej części stałych czytelników, postulujących powrót do poprzedniej struktury czasopisma (równy podział na część artykułową i informacyjną), a więc zwiększenie objętości informacji adresowanych do szerokiego kręgu odbiorców kosztem artykułów bardzo specjalistycznych, interesujących jedynie niewielkie liczbowo grupki naukowców. Spełnienie tego postulatu w obecnych warunkach nie jest jednak zadaniem łatwym. O wiele prościej jest bowiem uzyskać opracowanie o charakterze naukowym, którego opublikowaniem jest z wielu względów bardzo zainteresowany autor, niż nawet drobną, ale bardzo użyteczną dla wszystkich informację. Już od dłuższego czasu zespół redakcyjny odczuwa skutki wzrastającej nierównowagi na informatycznym rynku pracy powstałej w wyniku znacznego odpływu specjalistów poza granice kraju, jak i wzmożonego ich zapotrzebowania ze strony krajowych użytkowników. Na miejsce wyjeżdżających za granicę jest coraz trudniej pozyskać nowych współpracowników redakcji. Mimo znacznego finansowego usamodzielnienia redakcji w ramach nowej formuły organizacyjnej Wydawnictwa SIGMA (od 1.1.1990 r. działa jako Spółka z o.o.) wydaje się, że we wspomnianej sytuacji na rynku pracy będzie nam trudno stworzyć dostatecznie konkurencyjne warunki finansowe nie tyko dla pozyskania stałych współpracowników, ale nawet autorów takich tematów, na które istnieje rzeczywiste zapotrzebowanie.

W tej sytuacji pozostaje nam tylko zwrócenie się do naszych Czytelników z apelem o zgłaszanie chęci współpracy z redakcją. Z podobnym apelem zwracaliśmy się wielokrotnie przed dziesięcioma laty, pozyskując tą drogą wielu cennych współpracowników. Dziś, jako naturalna reakcja na nadużywanie tego rodzaju metod oddziaływania, coraz powszechniejsze są niestety postawy wyłącznie komercyjne. Znaczny wzrost liczebności środowiska stwarza jednak realną nadzieję, że znajdzie się choćby niewielka cząstka tego środowiska, która i tym razem pozytywnie zareaguje na nasz apel. Liczymy zwłaszcza na zgłoszenia osób młodych, wykształconych przez wyższe uczelnie w sposób profesjonalny, którzy swą wiedzę, tak istotną przecież dla gospodarczego przyspieszenia rozwoju kraju, zdecydują się z naszą pomocą przekazywać innym.

Redakcja

cz. 18.576

Pierwszą wersję tego artykułu przedstawiono na VII Międzynarodowej Konferencji „Expert Systems and Their Applications", która odbyła się w Avignon, w dniach 11–15 maja 1987 roku.

MAREK HOŁYŃSKI
Boston University
i Massachusetts Institute of Technology
Boston/Cambridge, stan Massachusetts
USA

Sprzęg graficzny jako system ekspertowy

Systemy ekspertowe i grafika komputerowa mają wiele wspólnego i wydaje się, że ich połączenie będzie procesem dość naturalnym. Techniki stosowane w systemach ekspertowych są bardzo użyteczne w wielu zastosowaniach grafiki komputerowej. Z drugiej strony, efektywność systemów ekspertowych zależy coraz bardziej od sprzężenia wizualnego, wejścia obrazowego i graficznego przedstawienia wyników. Ponieważ eksperci omawiają problemy i ilustrują ich rozwiązania za pomocą obrazów (rysunków), skomputeryzowane systemy ekspertowe powinny czynić tak samo.

Nowa generacja systemów graficznych umożliwia konstruowanie coraz inteligentniejszych sprzęgów człowiek-komputer i dostarcza lepszych technik do zarządzania tą współpracą. Systemy nowej generacji automatycznie decydują o wyborze najlepszej formy prezentacji zbioru danych, dostosowanej do wymagań indywidualnych użytkowników. Takie decyzje, oparte na właściwościach danych, celach użytkowników, zasadach projektowych i regułach percepcji, wymagają użycia technik sztucznej inteligencji, które ułatwiają organizowanie określonej wiedzy graficznej i operowania nią.

Do technik sztucznej inteligencji, które mogą być implementowane w sprzęgach graficznych, należą: rozwiązywanie problemów, podejmowanie decyzji, rozumowanie wnioskujące, nabywanie wiedzy i maszynowe uczenie się. Umożliwiają one zapewnienie systemowi zdolności do wytwarzania reguł obrazowania (przez uogólnienie przykładów wizualnych dostarczonych przez użytkownika), do udoskonalania ich i rozszerzania na nieprzewidziane okoliczności oraz do uczenia się na podstawie doświadczenia. Sprzęgi wyposażone w te właściwości stanowią faktycznie pierwszy krok do tworzenia graficznych systemów ekspertowych. Graficzne systemy ekspertowe powinny zapewniać możliwość analizowania, tworzenia i modyfikowania obrazów komputerowych zgodnie z preferencją obserwatora, biorąc pod uwagę całą informację kontekstową.

Do grupy systemów, w których skutecznie zastosowano tę metodykę, należy sytem APEX (ang. *Automatic Pictorial EXplanations*) opracowany w Brown University [2]. APEX generuje obrazy zgodnie z działaniem rozwiązywacza problemów dla akcji fizycznych w świecie trójwymiarowym. Zawiera reguły automatycznego określania obiektów, które mogą być pokazane na obrazie, stylu i poziomu reprezentowania szczegółów, które powinny być odtworzone, metody służącej do przedstawienia samej akcji i specyfikacji obrazu.

Interesujące badania nt. rozumienia, rozpoznawania i sprawdzania właściwości wizualnych wysokiego poziomu przeprowadzono w laboratorium Sztucznej Inteligencji MIT [11]. Z kolei, na Uniwesytecie Harwardzkim [3] opracowano i zrealizowano technikę automatycznej syntezy opisu obiektów graficznych na podstawie specyfikacji wysokiego poziomu oraz generowania różnych obrazów dla różnych obserwatorów i zadań. Innym przykładem z Laboratorium Sztucznej Inteligencji MIT [1] jest system PPS (ang. *Primitive Presentation System*), zdolny do wytworzenia tzw. prezentacyjnej bazy danych (bezpośrednio obserwowalnej) na podstawie aplikacyjnej bazy danych, za pomocą edytora prezentacyjnego (tj. edytora tekstowo-obrazowego).

SYSTEMY Z BAZĄ WIEDZY I SPRZĘGI ADAPTACYJNE

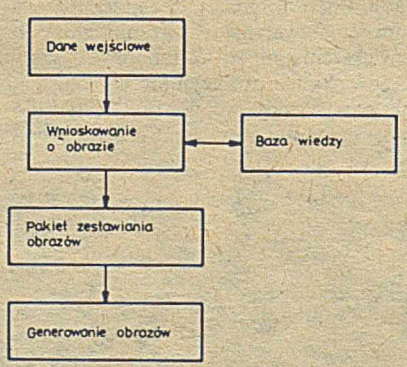
Istotną cechą wszystkich inteligentnych systemów graficznych jest ich zdolność do podejmowania decyzji, jak określone dane mają być przedstawione na obrazie. Pomijając różnice implementacyjne, tę właściwość można wyrazić za pomocą pakietu zestawiania obrazu, który generuje obraz na podstawie danych wejściowych i zastępuje całkowicie lub częściowo programy użytkowe oraz tradycyjne pakiety graficzne (rys.1).

Rys. 1. Tradycyjny i inteligentny system graficzny

Działanie pakietu zestawiania obrazu może być określone przez informację o znaczeniu samego obrazu lub za pomocą reguł wyświetlania ustalonych przez użytkownika. Dlatego dotychczasowy rozwój graficznych systemów ekspertowych następował głównie w dwóch kierunkach:
– systemy graficzne z bazą wiedzy, które wiążą reprezentację graficzną obrazu z jego znaczeniem,
– adaptacyjne sprzęgi graficzne, w których algorytmy generowania obrazów obejmują ograniczone dziedziny zastosowań, wiedzę użytkownika i preferencje wizualne.

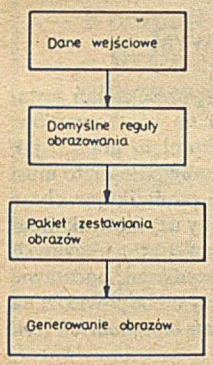
Ponieważ preferencje wizualne są ściśle powiązane ze znaczeniem, rozróżnienie tych dwóch kierunków nie zawsze jest łatwe i w rzeczywistych systemach często występują elementy obu podejść. W wielu wypadkach ta klasyfikacja nie zawodzi i ułatwia rozróżnienie celów odpowiednich systemów.

Do efektywnej wizualizacji danych system graficzny powinien mieć pewną wiedzę o dziedzinie zastosowań. Ponadto, musi umożliwiać użytkownikowi wprowadzanie w prosty sposób jego własnej wiedzy (najlepiej wizualnie) i zawierać pewne mechanizmy wnioskowania o obrazie (rys. 2). Taki system powinien więc składać się z narzędzi zarówno do reprezentowania wiedzy, jak i do przedstawiania grafiki. Ten rodzaj grafiki komputerowej wrażliwej na kontekst zaczyna być obecnie stosowany w systemach prezentacji, wyświetlaczach ikonicznych, sprzęgach bez danych i animowaniu obrazów.



Rys. 2. System graficzny z bazą wiedzy

Systemy graficzne mogą być niezależne od określonej dziedziny zastosowań lub od poszczególnych użytkowników. Reprezentacja graficzna powinna być dostosowana do rodzaju użytkownika i istoty zastosowania. Sposób w jaki architekt odbiera obraz budynku jest zasadniczo odmienny od sposobu w jaki na tę samą strukturę patrzy specjalista inżynierii lądowej lub artysta. W celu dostosowania reprezentacji danych do preferencji, potrzeb i wiedzy osób, które będą z niej korzystać, używa się adaptacyjnych sprzęgów graficznych, opartych na zbiorze standardowych reguł wyświetlania, ukierunkowanych na użytkownika (rys. 3).



Rys. 3. Adaptacyjny sprzęg graficzny

Poniżej omówię dwa przykłady oprogramowania odpowiadającego dwóm głównym rodzajom inteligentnych systemów grafiki komputerowej. Ilustracją systemu graficznego z bazą wiedzy jest SNEPS (ang. *Semantic Network Processing System*) zintegrowany z pakietem Graflisp, a przykładem adaptacyjnego sprzęgu graficznego jest AGA (ang. *Adaptive Graphics Analyzer*).

SNEPS–GRAFLISP

SNEPS jest systemem interakcyjnym do tworzenia i eksploatowania ogólnej bazy wiedzy, opracowanym na uniwersytecie stanu Nowy Jork w Buffalo [13]. Implementację tego systemu do celów grafiki wykonano na Uniwesytecie Bostońskim [12]. Wiedza o systemie SNEPS jest reprezentowana jako etykietowany graf skierowany, w którym każdy węzeł przedstawia pojęcie. Krawędzie między węzłami grafu odpowiadają relacjom między tymi pojęciami. Znaczenie każdego węzła jest zawarte w strukturze całej sieci. Dzięki temu, możliwe jest przedstawienie dwóch różnych pojęć odnoszących się do tego samego obiektu graficznego, jak np. budynek (ang. *house*) i dom (ang. *home*).

Bardziej złożone i abstrakcyjne pojęcia mogą być definiowane za pomocą węzłów istniejących. Ta właściwość czyni SNEPS szczególnie użytecznym do reprezentowania pojęć graficznych, ponieważ wyrażanie obrazów i preferencje użytkowników wymagają abstrakcyjnego poziomu specyfikacji. Wiedza o tych konstrukcjach wysokiego poziomu może być tworzona przyrostowo, przy użyciu prostszych pojęć jako elementów składowych.

SNEPS umożliwia reprezentowanie w tej samej sieci stwierdzeń faktograficznych jak i reguł wnioskowania. Mechanizm systemu SNEPS pozwala na stosowanie reguł wnioskowania wyprzedzającego (ang. *forward*) i wstecznego (ang. *backward*), jak też rozumowania dwukierunkowego (ang. *bidirectional*). SNEPS ułatwia reprezentowanie różnych przekonań i preferencji różnorodnych użytkowników lub ich grup. Przykładowo, możliwa jest prezentacja następującego zdania: *„Architekci preferują umieszczanie ludzi na ekranie po prawej stronie budynku"*.

Graflisp jest pakietem graficznym zaprojektowanym w 1982 roku na Uniwersytecie Bostońskim, do zastosowań w sztucznej inteligencji [4]. Użyto w nim wielu technik sztucznej inteligencji do wykonywania złożonych operacji graficznych, jak np. dobieranie wzorców do wyrażania właściwości graficznych, specyfikowanie nazw kolorów, sprzęganie inteligentnych monitorów, i innych, jak obsługa błędów lub samomodyfikowanie kodu. Do reprezentowania złożonych relacji między obiektami w tym pakiecie zastosowano sieć opartą na hierarchicznej kracie. Dzięki temu możliwe jest przekazywanie odpowiedniej wiedzy o obiektach macierzystych (jak np. rozmiar, orientacja itp.) do obiektów potomnych, na zasadzie dziedziczenia w czasie ich powstawania.

Obiekty można tworzyć różnymi sposobami. Graflisp umożliwia definiowanie obiektów za pomocą danych (metoda standardowa) lub proceduralnie. Definiowanie danymi polega na ogół na opisie obiektów za pomocą współrzędnych punktów, jak np. tworzenie powierzchni obiektu na podstawie skończonej liczby wielokątów. Definiowanie proceduralne polega na opisie obiektów za pomocą funkcji, które określają obiekty przez ich typ. Oprócz definicji proceduralnych, takich jak sfery, powierzchnie rozwijania, oświetlenie itp., pakiet zawiera również funkcje umożliwiające dołączenie specjalizowanych typów obiektów opisanych przez użytkownika.

Graflisp jest niezależny od sprzętu. Działa na różnych komputerach (VAX 11/780, VAX 11/750, IBM 308ł i in.), w różnych systemach operacyjnych (Unix, VMS, VPS) i z różnymi translatorami Lispu (Franz Lisp, BU-Lisp i in.). Jest także niezależny od urządzeń zewnętrznych, tzn. może współpracować z różnymi urządzeniami graficznymi, np. AED-512, VT125, Tektronix 4013 i terminale kolorowe firmy Ramtek.

Wewnętrzna struktura sieci pakietu Graflisp czyni go szczególnie podatnym do połączenia z systemem SNEPS. Sprzęgając oba te narzędzia napisane w Lispie, stworzono system zdolny do kojarzenia abstrakcyjnych znaczeń i relacji między obiektami w hierarchicznej strukturze obrazu [6]. Celem tego połączenia było więc zbudowanie systemu, który umożliwia przyjmowanie abstrakcyjnych zapytań i określanie koncepcji sceny (złożonej ze zbioru składowych pojęć i relacji między nimi, odpowiadających aktualnemu zapytaniu), jak również opracowywanie specyfiki rzeczywistych obrazów.

Aby ocenić to podejście zdecydowano zbadać dziedzinę, w której należy generować obrazy tworząc je począwszy od poziomu konceptualnego. W tym celu wybrano rzeczywisty przykład, dostatecznie prosty, aby był możliwy do realizacji, lecz na tyle złożony, aby pozwalał na zweryfikowanie proponowanej metodyki [7].

Punkt wyjścia stanowiły pojęcia takie, jak: ludzie, domy, ludzie mieszkający w określonych domach i pracujący w określonych miejscach. Odpowiadając na przykładowe zapytanie *„Wyświetl mi Ronia i Biały Dom"*, system sam podejmuje decyzje co do skali poszczególnych elementów obrazu. Może na przykład zmniejszyć wielkość sylwetek ludzkich w porównaniu z Białym Domem i umieścić je w założonym miejscu. Może też podejmować bardziej złożone decyzje dotyczące położenia elementów na obrazie. Przykładowo, można zażądać pokazania sylwetek ludzkich przed Białym Domem, lecz na lewo od zwykłego budynku.

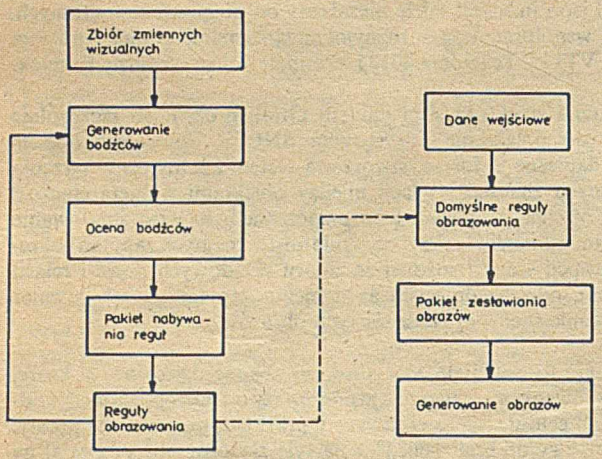
Oprócz tego można uwzględniać czasowy aspekt wnioskowania, posługując się zapytaniami o położenie osób względem ich domów lub miejsc pracy o określonej porze dnia. Przykładowo, po dodaniu do systemu reguły *„Ludzie znajdują się w miejscu pracy tylko w godzinach pracy, a poza tymi godzinami są w domu"*, system sam określa koncepcję według bieżącego czasu. W tej sytuacji, system będzie przekonany, że ludzie pracują podczas godzin pracy, a poza tym godzinami przebywają w domu (oczywiście, przekonania systemu nie muszą być zawsze zgodne z prawdą). Gdy zapytamy system o wyświetlenie domu Billa, powinien on najpierw sprawdzić czas i określić budynek będący tym domem. Jeżeli będą to godziny pracy, to system wyświetli zwykły dom wraz z obecną w tym domu Jane, ponieważ Bill jest w tym czasie w pracy. Podczas weekendu lub poza godzinami pracy, pokazani zostaną oboje, Jane i Bill, w swoim domu.

ADAPTACYJNY ANALIZATOR GRAFICZNY

Adaptacyjny analizator graficzny AGA stworzono na Uniwersytecie Bostońskim, aby umożliwić ocenianie preferencji wizualnych użytkowników i włączanie ich jako reguł wyświetlania do systemu graficznego. Takie reguły efektywnej prezentacji wizualnej danych opracowano doświadczalnie, wytwarzając szereg obrazów za pomocą specjalnego programu sterowanego przez menu [8] i korelując je z oceną obserwatorów za pomocą programu automatycznego rozpoznawania reguł. Reguły można wykorzystać jako standardowe w adaptacyjnym systemie graficznym, który generuje, zmienia i udoskonala obrazy infrakcyjne (rys. 4).

Użyty tu program automatycznego rozpoznawania reguł opracowano na Uniwersytecie stanu Illinois w Urbana-Champaign. Reprezentuje on wiedzę za pomocą reguł działania warunkowego, których format jest oparty na rachunku logiki zmiennowartościowej, będącej rozszerzeniem

logiki wielowartościowej o zmienne utypizowane. Program ujawnia strukturalne opisy klas obiektów, tzn. opisy określające nie tylko atrybuty obiektów, lecz także relacje między składowymi i podskładowymi tych obiektów. Podobne programy sprawdzono w wielu zastosowaniach praktycznych. Dobrze znane jest użycie indukcyjnego programu AQ11, uczącego się na podstawie przykładów, do generowania reguł diagnozowania chorób soi [10]. Programy indukcyjne stosowano również skutecznie do wytwarzania reguł opartych na sygnałach sonaru, do rozróżniania ryb od kry podwodnej. Uzyskane wyniki wykazały, że w rozwiązaniu specyficznych i dobrze określonych problemów indukcyjne metody uczenia się są już dostatecznie skuteczne, aby ujawniać użyteczną wiedzę. Niniejsze badania autora również potwierdzają tę tezę.



Rys. 4. Generowanie domyślnych reguł obrazowania

W przykładzie generowano obrazy testowe jako abstrakcyjne macierze, złożone z kontrolowanych kombinacji 36 elementów podstawowych. Zbiór atrybutów użyty do wytworzenia tych obrazów wzorcowych obejmował: złożoność, regularność, symetrię, równowagę, różnorodność, zajętość i kolor. Dwieście osób badanych poproszono o ocenę obrazów ilustrujących te zmienne, według czterostopniowej skali preferencji (ocena wysoka, średniowysoka, średnioniska i niska). Wyniki tej oceny wprowadzono do programu automatycznego rozpoznawania reguł, w sposób przedstawiony w tabeli.

Przykład danych dla programu automatycznego rozpoznawania reguł (1 – równowaga, 2 – różnorodność, 3 – złożoność, 4 – regularność, 5 – symetria, 6 – zajętość, 7 – kolor, m – mała, ś – średnia, d – duża, b – biały, n – niebieski, z – zielony).

Atrakcyjność	Atrybuty						
	1	2	3	4	5	6	7
Niska	ś	m	m	ś	m	m	b
Średnioniska	d	m	ś	m	ś	m	z
Średniowysoka	ś	m	m	ś	m	d	z
Wysoka	ś	m	d	m	d	d	n

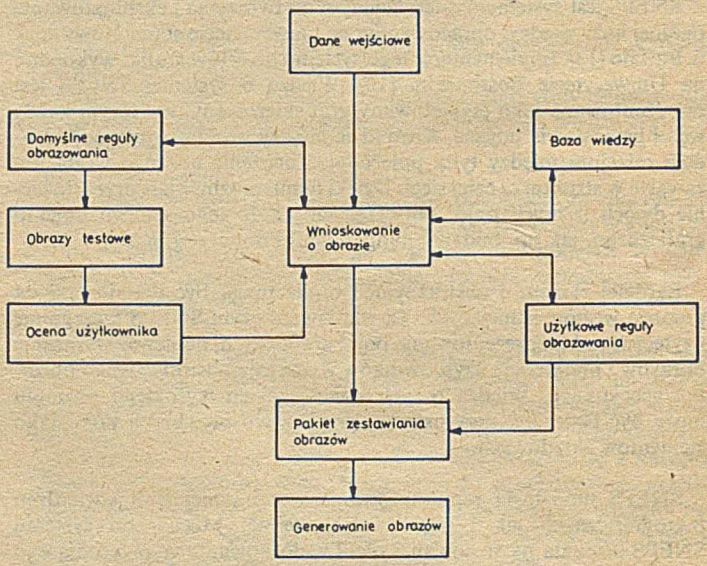
Po użyciu programu rozpoznawania reguł dla danych uzyskanych za pomocą programu sterowanego przez menu, uzyskano bardzo jasne reguły prezentacji dla określonego zbioru obserwatorów. Tymi regułami może obecnie posługiwać się system graficzny do generowania, zmieniania i udoskonalania obrazów, aby dostosować je do preferencji obserwatorów. Poniżej podano dwa przykłady otrzymanych reguł.
Atrakcyjność jest niska, jeśli złożoność = *m*, zajętość = *m* i kolor = *b* lub *z*.
Atrakcyjność jest wysoka, jeżeli regularność = *ś* lub *d*, symetria = *d* i kolor = *n* lub jeżeli równowaga = *ś* lub *m*, regularność = *d* i kolor = *z* lub *n*.

W następnej fazie projektu te reguły prezentacji przyjęto jako standardowe reguły wyświetlania przy tworzeniu obrazów. W tym wypadku jednak, wyposażono sprzęg w zdolność modyfikowania reguł przez specyficzną wiedzę pochodzącą od poszczególnych użytkowników, dotyczącą ich osobistych sądów i niepowtarzalnych właściwości. Obecna wersja systemu AGA wyświetla ciąg obrazów na podstawie danych standardowych, pozwalając użytkownikom na wyrażenie osobistych preferencji. Opierając się na preferowanych wyborach użytkowników, dzięki mechanizmowi poprawiania wiedzy system modyfikuje dane standardowe i wytwarza nowe, odpowiadające im efekty wizualne. Ten proces można powtarzać kilkakrotnie, aż do otrzymania takich wartości atrybutów, które użytkownik będzie przedkładał ponad wszystkie poprzednie wybory. Tak skonstruowany zbiór reguł wyświetlania można stosować do tworzenia obrazów wyświetlanych dla tego szczególnego użytkownika.

KU GRAFICZNYM SYSTEMOM EKSPERTOWYM

W dotychczasowych systemach graficznych dużą część zadań zestawiania sceny i tworzenia obiektów przenoszono na użytkowników. Wymagano, aby użytkownik był zarówno projektantem grafiki, jak i programistą. Nawet jeśli użytkownik ma obie te umiejętności, to musi spędzać całe godziny przed ekranem, nieustannie modyfikując obraz tak, aby wyglądał poprawnie. Idealnie byłoby, gdyby użytkownik mógł mówić do komputera, używając znaczących lub nawet abstrakcyjnych terminów, a komputer rozumiał to i składał sceny na podstawie znaczenia obrazów. Jednakże, taka możliwość zależy od odpowiedzi na pytanie: *„Czy komputery mogą wnioskować o składzie oryginalnych scen i używać inteligencji do tworzenia obrazów?"*

Inteligentne systemy graficzne, które dotąd opracowano, stanowią potencjalną odpowiedź pozytywną. Wiele z nich jest w rzeczywistości pierwszym krokiem na drodze do stworzenia graficznych systemów ekspertowych. Systemy te dadzą nam możliwość analizowania, tworzenia i modyfikowania obrazów komputerowych z punktu widzenia ich rozumienia, wskutek powiązania znaczenia obrazu z jego reprezentacją graficzną (rys. 5). Staną się nie tylko doskonałym narzędziem do automatycznego projektowania obrazów, lecz będą umożliwiały wyjaśnienie, dlaczego wybrano akurat ten szczególny sposób prezentacji.



Rys. 5. Graficzny system eksportowy

Immanentną cechą graficznego systemu ekspertowego powinna być zdolność do tworzenia nowych hierarchii obiektów, realizowania zależności semantycznych między obiektami, zestawiania scen oryginalnych i budowania – w postaci obrazu – sceny określonej na poziomie konceptualnym. Ważna właściwość takich systemów polega na tym, że w procesie realizowania i zestawiania wykorzystuje się wiedzę zarówno o ogólnych ludzkich preferencjach wizualnych, jak i osobistych preferencjach użytkowników. W połączeniu ze zdolnością systemu do reprezentowania przekonań indywidualnych obok znanych faktów, ta właściwość stanowi mocną podstawę do nowoczesnych zastosowań grafiki komputerowej.

Tłumaczył i opracował
JANUSZ ZALEWSKI

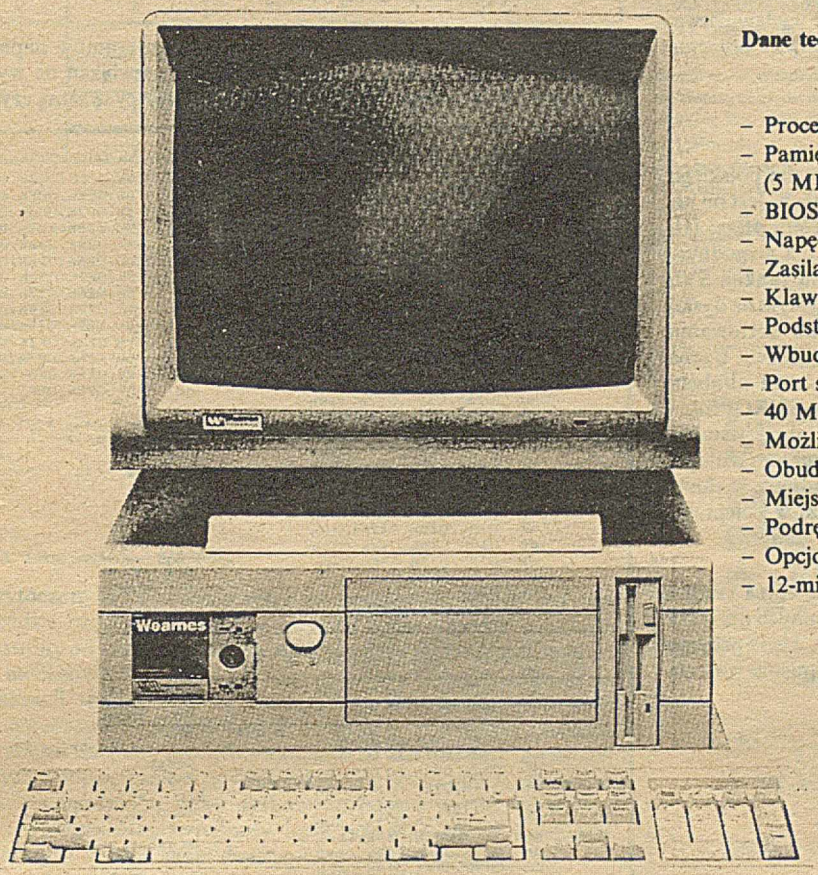
dokończenie na s. 7

KOMPUTER
Z WBUDOWANĄ PRZYSZŁOŚCIĄ

Komputer zaprojektowany przez czołowego światowego producenta mikrokomputerów – firmę ADVANCED LOGIC RESEARCH i wyprodukowany przez firmę WEARNES TECHNOLOGY.

Komputer, którego możliwości i cena oszałamiają konkurencję.

Komputer, który nie zestarzeje się, dzięki możliwości rozbudowy do architektury 386SX i 486.



Dane techniczne:

SERIA WEARNES BOLDLINE „M"

- Procesor 80286 – 12.5 Mhz
- Pamięć 1 MB RAM, możliwość rozbudowy do 16 MB (5 MB na płycie głównej)
- BIOS Phoenix
- Napęd dyskietek 3,5" 1,44 MB
- Zasilacz 110 Watt
- Klawiatura 101 klawiszy
- Podstawka dla koprocesora matematycznego 80287
- Wbudowany sterownik dyskowy z przeplotem 1:1
- Port szeregowy i równoległy
- 40 MB dysk sztywny
- Możliwość korzystania z EMS 4.0
- Obudowa typu „compact"
- Miejsce na dwa napędy 5,25" o wysokości 1/2
- Podręcznik i dyskietka z programem konfiguracyjnym
- Opcjonalna rozbudowa do 386SX i 486
- 12-miesięczna gwarancja

Komputer z wbudowaną przyszłością!

BOLDLINE COMPUTERS – GRUPA MICOMP
Biuro Informacji Techniczno-Handlowej
ul. Astrów 7, 40–045 Katowice
tel. 51-86-28 (telefaks), teleks 315687 COMP PL

EO/1123/89

ANDRZEJ DYDEK
Warszawa

Porządkowanie zbiorów danych według polskiego alfabetu

W różnych zestawach polskich znaków dla mikrokomputerów oferowanych przez krajowe firmy przyjmuje się różne kody znaków diakrytycznych właściwych dla polskiego alfabetu. Obecnie użytkownik żąda (i słusznie!) stosowania jednolitych zasad zarówno przy korzystaniu z klawiatury, jak i z drukarki. Brak jakiejkolwiek polskiej normy w powyższej kwestii nie ułatwia życia użytkownikom i producentom, a nawet chaos ten jeszcze pogłębia. Jednakże, jak się wydaje, decyzja normalizacyjna została już podjęta i to poza granicami kraju. Otóż, jak podano oficjalnie, firma IBM nie wycofała się z poprzednio podjętych decyzji i dla polskich znaków diakrytycznych pozostawiła kody ustalone kilka lat temu w tablicy Latin 2. Listę kodów odpowiadających polskim literom przedstawiono w tabeli 1, na podstawie wydawnictwa: Registry – Graphic Character Sets and Code Pages. IBM Corporate Specification, Draft CH-3-3220-050, June 1986[1]).

ZASADY PORZĄDKOWANIA

Problem korzystania ze znaków charakterystycznych dla polskiego alfabetu występuje coraz ostrzej przy pracy z bazami danych. Jest on nie rozwiązany do dziś dla licznych użytkowników interpretera dBase III, natomiast użytkownicy systemu FoxBase dysponują już wygodnym narzędziem. Przyjęte rozwiązanie polega na stosowaniu funkcji SYS w powiązaniu z plikiem EUROPEAN.MEM przechowującym na dysku zmienne robocze. Funkcja ta, przy indeksowaniu lub sortowaniu umożliwia wymianę określonych znaków na inne. W tym celu trzeba jedynie pozlepiać znaki alfabetu narodowego ze znakami alfabetu łacińskiego. Przy takim sposobie zdefiniowania zmiennej EUROPEAN zawartej w pliku dyskowym EUROPEAN.MEM wystarczy wstawić jako bajt 165 literę *a*, po to aby litery *ą* i *a* były nierozróżnialne w indeksowaniu i sortowaniu. Oznacza to jednak przykładowo, że słowa *LOS* i *ŁOŚ* pojawią się wówczas w pliku posortowanym w kolejności zależnej jedynie od ich pierwotnej kolejności w zbiorze danych.

Jak rozwiązać ten ostatni problem? Ponieważ użytkownika rzadko interesuje sortowanie zbioru danych względem znaków innych niż litery (niekiedy wchodzą w grę dodatkowo cyfry i znaki spacji), więc z pełnego zestawu 256 kodów można wybrać odpowiednie do zastąpienia w sumie około 100 znaków.

```
function plsrt
parameters x
private i,y
if type('x')='C'
  y=''
  for i=1 to len(x)
    y=y+substr(pl_sort.asc(substr(x,i,1)),1)
  next
  return y
endif
return x
```

Wydruk 1. Tekst programu PLSORT.PRG

Nic nie stoi na przeszkodzie, aby zastosować zaprezentowany schemat także w programach systemu dBase III kompilowanych przez Clipper. Wystarczy jedynie do katalogu Clippera dołączyć plik PLSORT.MEM zawierający tabelę konwersji przypisaną zmiennej. Oprócz pliku PLSORT.MEM w tym samym katalogu musi także znajdować się kod pośredni (w pliku OBJ) wygenerowany na podstawie kodu źródłowego przedstawionego na wydruku 1. Odpowiedni kod pośredni można uzyskać po wydaniu z poziomu DOS polecenia:

CLIPPER PLSORT

Naturalnie, w programie korzystającym z przedstawionej funkcji PLSORT, przy konsolidacji należy także podać jako parametr FI identyfikator pliku PLSORT.OBJ.

Clipper umożliwia również stosowanie podprogramów napisanych w języku asemblera. Dla zwolenników takich rozwiązań na wydruku 2 przedstawiono podprogram funkcyjny wykonujący tę samą czynność.

```
PUBLIC SRTPL
extrn   __PARC:far
extrn   __RETC:far
_PROG   SEGMENT 'CODE'
        ASSUME CS:_PROG
SRTPL   PROC  FAR
        jmp aaa

buf:    db 255 dup(?)
tabl:   db  0
        db  001h,002h,003h,004h
        db  005h,006h,007h,008h
        db  009h,00Ah,00Bh,00Ch
        db  00Dh,00Eh,00Fh,010h
        db  011h,012h,013h,014h
        db  015h,016h,017h,018h
        db  019h,01Ah,01Bh,01Ch
        db  01Dh,01Eh,01Fh,020h
        db  021h,022h,023h,024h
        db  025h,026h,027h,028h
        db  029h,02Ah,02Bh,02Ch
        db  02Dh,02Eh,02Fh,030h
        db  031h,032h,033h,034h
        db  035h,036h,037h,038h
        db  039h,03Ah,03Bh,03Ch
        db  03Dh,03Eh,03Fh,040h
        db  040h,042h,043h,045h
        db  046h,048h,049h,04Ah
        db  04Bh,04Ch,04Dh,04Eh
        db  050h,051h,053h,055h
        db  054h,056h,057h,059h
        db  05Ah,05Bh,05Bh,05Dh
        db  05Ch,05Dh,05Bh,05Ch
        db  05Dh,05Eh,05Fh,060h
        db  060h,062h,063h,065h
        db  066h,068h,069h,06Ah
        db  06Bh,06Ch,06Dh,06Eh
        db  070h,071h,073h,075h
        db  074h,076h,077h,079h
        db  07Ah,07Bh,07Bh,07Dh
        db  07Ch,07Dh,07Bh,07Ch
        db  07Dh,07Eh,07Fh,080h
        db  081h,082h,083h,084h
        db  085h,064h,087h,06Fh
        db  089h,08Ah,08Bh,08Ch
        db  05Eh,08Eh,044h,090h
        db  091h,092h,093h,094h
        db  095h,096h,058h,078h
        db  099h,09Ah,09Bh,09Ch
        db  04Fh,09Eh,09Fh,0A0h
        db  0A1h,074h,0A3h,041h
        db  061h,0A6h,0A7h,047h
        db  067h,0AAh,07Eh,0ACh
        db  0ADh,0AEh,0AFh,0B0h
        db  0B1h,0B2h,0B3h,0B4h
        db  0B5h,0B6h,0B7h,0B8h
        db  0B9h,0BAh,0BBh,0BCh
        db  05Fh,07Fh,0BFh,0C0h
        db  0C1h,0C2h,0C3h,0C4h
        db  0C5h,0C6h,0C7h,0C8h
        db  0C9h,0CAh,0CBh,0CCh
        db  0CDh,0CEh,0CFh,0D0h
        db  0D1h,0D2h,0D3h,0D4h
        db  0D5h,0D6h,0D7h,0D8h
        db  0D9h,0DAh,0DBh,0DCh
        db  0DDh,0DEh,0DFh,054h
        db  0E1h,0E2h,052h,072h
        db  0E5h,0E6h,0E7h,052h
        db  055h,072h,0EBh,079h
        db  0EDh,0EEh,0EFh,0F0h
        db  0F1h,0F2h,0F3h,0F4h
        db  0F5h,0F6h,0F7h,0F8h
        db  0F9h,0FAh,0FBh,0FCh
        db  0FDh,0FEh,0FFh

aaa:    push bp
        mov  bp,sp
        push ds
        push es
        push si
        push di
        mov  ax,1
        push ax
        call __PARC
        add  sp,2
        mov  ds,dx
        mov  si,ax
        mov  ax,cs
        mov  es,ax
        lea  di,buf
        mov  cx,254
        cld
        lea  bx,tabl
        xor  ah,ah
a1:     lodsb
        xlat byte ptr cs:[bx]
        stosb
        and  al,al
        jz   a2
        loop a1
        xor  al,al
        stosb
a2:     mov  dx,cs
        lea  ax,buf
        pop  di
        pop  si
        pop  es
        pop  ds
        pop  bp
        push dx
        push ax
        call __RETC
        add  sp,4
        ret
SRTPL   ENDP

_PROG   ENDS
        END
```

Wydruk 2. Tekst programu SORTPL.ASM

[1]) Należy dodać, że specyfikacja IBM jest sprzeczna z aktualnie obowiązującą normą ISO 8859-2 (przyp. red.).

Tabela 1. Kody polskich liter w tablicy IBM Latin 2

ą	ć	ę	ł	ń	ó	ś	ź	ż	Ą	Ć	Ę	Ł	Ń	Ó	Ś	Ź	Ż
165	134	169	136	228	162	152	171	190	164	143	168	157	227	224	151	141	189

WYNIKI TESTÓW

Różne warianty rozwiązania zadania testowano na mikrokomputerze EMIX 86 XT Turbo za pomocą programu przedstawionego na wydruku, skompilowanego przez Clipper w wersji Summer'87 oraz skonsolidowanego przez program Plink 86 w wersji 2.24. W tabeli 2 przestawiono wyniki działania programu testującego dla różnych wielkości pliku XXX.DBF oraz dla różnych długości pola NAZWA. W kolumnie Indeks 1 przedstawiono wyniki przy indeksowaniu pliku względem pola NAZWA, w kolumnie Indeks 2 – dla indeksowania względem funkcji napisanej w języku asemblera SRTPL(NAZWA), natomiast w kolumnie Indeks 3 – dla indeksowania względem funkcji napisanej w języku Clippera PLSRT(NAZWA).

Tabela 2. Działanie programu testującego

Długość hasła w znakach	Liczba rekordów w tysiącach	Czas realizacji w sekundach		
		Indeks 1	Indeks 2	Indeks 3
10	1	8	10	90
	2	17	20	180
	4	34	47	363
	8	75	90	734
20	1	11	11	161
	2	19	21	321
	4	44	51	647
	8	83	99	–
50	1	12	14	373
	2	27	31	749
	4	54	64	1500
	8	109	126	–

```
use xxx
bt=time()
index on nazwa to xx1.tmp
et=time()
? elaptime(bt,et)
use
use xxx
bt=time()
index on srtpl(nazwa) to xx2.tmp
et=time()
? elaptime(bt,et)
use
use xxx
bt=time()
restore from plsort additive
index on plsrt(nazwa) to xx3.tmp
et=time()
? elaptime(bt,et)
use
quit
```

Wydruk 3. Tekst programu TEST.PRG

Z przytoczonych rezultatów wynika, że czas wykonania operacji w przypadku funkcji napisanej w języku Clippera zależy w sposób bardzo istotny od rozmiaru danych, natomiast dla pozostałych obydwu operacji indeksowania rozmiar danych nie wpływa tak zdecydowanie na czas realizacji. Należy jednak pamiętać, że stosowany zazwyczaj schemat indeksowania zbioru danych wymaga najpierw indeksowania pustego zbioru danych i modyfikowania go na bieżąco przy zmianach związanych z dopisywaniem nowych rekordów oraz korektą rekordów zapisanych uprzednio w pliku dyskowym. Istotne jest to, że w tym wypadku w bazie danych rekordy będą pojawiały się w kolejności uwzględniającej polskie znaki diakrytyczne (alfabetycznie).

Uprzejmie informujemy, że od stycznia 1990 r. wszelkie sprawy przyjmowania ogłoszeń do INFORMATYKI załatwiane są bezpośrednio w Redakcji.

Sprzęg graficzny jako system ekspertowy

dokończenie ze s. 4

LITERATURA

[1] Ciccarelli E. C.: Presentation Based User Interfaces. Technical Report 794, MIT Artificial Intelligence Lab., Cambridge (MA), 1984
[2] Feiner S.: Apex – An Experiment in the Automated Creation of Pictorial Explanations. Computer Graphics and Applications, pp. 29–37, November 1985
[3] Friedell M.: Automatic Synthesis of Graphical Object Description. Computer Graphics, pp. 53–62, July 1984
[4] Gardner B. R.: GRAFLISP – A graphics package design for artificial intelligence applications. M. Sc. Thesis, Dept. of Computer Science. Boston University, 1985
[5] Hołyński M.: Knowledge-Based Computer Graphics – Intersections of AI and Computer Graphics. Siggraph '86 Panel, Dallas (TX), August 1986
[6] Hołyński M., Gardner B. R., Ostrovsky R.: Knowledge-Based Generation of Computer Images. Proc. Computer Graphics '86, National Computer Graphics Association, Anaheim (CA), May 1986
[7] Hołyński M., Gardner B. R., Ostrovsky R.: Utilisation d'Informations Semantiques en Synthese d'Images. Deuxieme Colloque Image, pp. 627–632, Nice (France), April 1986
[8] Hołyński M., Garneau R., Lewis E.: An Adaptive Graphics Interface for Effective Visual Representation. Proc. Eurographics '86, pp. 195–206, Lisbon (Portugal), August 1986
[9] Hołyński M., Lewis E.: Effective visual representation of computer generated. Proc. 5th Symp. on Small Computers in the Arts. Pp. 9–13, IEEE Computer Society Press, New York (NY), October 1985
[10] Michalski R. S., Chilauski R. L.: Learning by being told and learning from examples – an experimental comparison of the two methods of knowledge acquisition on the context of developing an expert system for soybean disease diagnosis. Int. Policy Analysis and Information Systems, Vol. 4, No. 2, pp. 125–161, 1980
[11] Montalvo F. S.: Diagram Understanding – The Intersection of Computer Vision and Graphics. Memo 873, MIT Artificial Intelligence Lab., Cambridge (MA), 1985
[12] Ostrovsky R., Gardner B. R., Hołyński M.: Semantic Network Reasoning for Picture Composition. Proc. Graphics Interface '86, pp. 20–25, Vancouver (Canada), May 1986
[13] Shapiro S.: The SNePS Semantic Network Processing System. N. V. Findler (Ed.), Associative Networks, pp. 179–203, Academic Press, New York, 1979.

Kombinat Górniczo-Hutniczy Miedzi

ZAKŁAD BUDOWNICTWA GÓRNICZO-HUTNICZEGO w Lubinie

ul. M. C. Skłodowskiej 92

zatrudni natychmiast

INŻYNIERA INFORMATYKA z praktyką.

Zakład zapewnia:

- otrzymanie w krótkim czasie mieszkania
- uzyskanie wysokiego wynagrodzenia wg Układu Zbiorowego Pracy dla Górnictwa Rud oraz realizację przywilejów wynikających z Karty Górnika.

Zgłoszenia przyjmuje i udziela informacji DZIAŁ KADR i SZKOLENIA ZAWODOWEGO, tel. 463-220, ul. M. C. Skłodowskiej 92, 59–300 Lubin.

EO/739/89

KaNet

Przedsiębiorstwo Techniczno-Handlowe

„KaNet" Sp. z o.o.

40–168 Katowice, ul. Kaktusów 7
tel. 583-613, teleks 312627 kanet pl

oferuje:

własne oprogramowanie narzędziowe pracujące pod kontrolą systemów operacyjnych UNIX V, XENIX 286,386, Advanced Net Ware (NOVELL), MS DOS

MIXEDIT

pełnoekranowy edytor tekstowy, rozszerzony odpowiednik STANDARDU NORTON EDYTORA, umożliwiający m.in.:

- edycję w dwóch oknach wielu plików,
- wybór pliku do edycji z listy plików,
- realizację operacji na blokach tekstu,
- realizację operacji poszukiwania,
- wizualizację tekstu w postaci skondensowanej,
- wymianę fragmentów tekstu,
- definiowanie i przechowywanie na dysku do 9 makrodefinicji,
- dynamiczne definiowanie parametrów pracy.

MIXCOMM

program wspomagający zarządzanie systemem plików – rozszerzony odpowiednik STANDARDU NORTON COMMANDERA, umożliwiający m.in.:

- wyświetlenie w jednym lub dwu oknach informacji o zawartości katalogów systemu,
- operacje na plikach w katalogach,
- edycję lub przeglądanie plików,
- wyświetlanie podstawowych informacji o użytkowniku,
- wykonywanie w dowolnej chwili poleceń systemowych (SHELL),
- korzystanie z informacji pomocniczej (HELP),
- sterowanie poprzez „hotkeys" i rozwijanie „menu".

Programy obsługują TERMINALE STANDARDU VT 100 i ANST.

„KaNet" dostarcza dokumentację w języku polskim, a na życzenie program demo po kosztach nośnika magnetycznego.

EO/1108/89

ANDRZEJ PAPROCKI
Instytut Maszyn Matematycznych
Warszawa

Problemy implementacji języka Ada

Próby implementacji Ady rozpoczęły się jeszcze przed ogłoszeniem oficjalnej normy. Początkowo realizowano poszczególne konstrukcje i był to okres burzliwego rozwoju różnych podzbiorów Ady. Próby te z jednej strony wpłynęły na ostateczną postać normy w ten sposób, że niektóre konstrukcje zawarte we wstępnej propozycji zostały usunięte lub zmodyfikowane, z drugiej strony zaś, udowodniły możliwość efektywnej realizacji języka. Miało to istotne znaczenie, gdyż przez pewien czas istniała obawa, że Ada może podzielić los Algolu 68, który był językiem niezwykle bogatym, zawierającym wiele nowatorskich rozwiązań, ale nie można go było efektywnie zaimplementować.

Ogłoszenie na początku 1983 roku ostatecznej normy języka, przyznanie nazwie Ada statusu znaku handlowego i rozwinięcie przez Departament Ochrony USA prac nad procedurą legalizacyjną kompilatorów zakończyło fazę powstawania kolejnych podzbiorów; od tego momentu zaczęły powstawać kompilatory pełnej Ady. Za kompilator Ady uznawany jest jedynie kompilator, który pomyślnie przeszedł proces legalizacji i otrzymał atest. Pierwszym legalizowanym translatorem Ady był interpreter opracowany w New York University. Otrzymał on atest w marcu 1983 roku. Początkowo kompilatory powstawały przede wszystkim na zlecenie Armii USA. Obecnie istnieje już kilkadziesiąt legalizowanych kompilatorów Ady, działających na wszystkich rodzajach większych komputerów, przy czym najwięcej ich jest dla komputerów VAX. Istnieją także dwa legalizowane kompilatory Ady dla IBM PC: jeden firmy Alsys dla IBM PC/AT ze specjalnym rozszerzeniem sprzętowym w postaci dodatkowej płyty pamięci, drugi firmy Meridian o nazwie AdaVantage dla IBM PC/XT z pamięcią o pojemności co najmniej 640 KB.

Zarówno kompilatory Ady, jak i wszystkie produkty z nią związane są uznawane przez rząd USA za technologie militarne i jako takie zostały umieszczone na liście embargowej, dlatego nie ma możliwości oficjalnego zakupienia tych produktów przez kraje Europy Wschodniej. W krajach RWPG podjęto wiele prób stworzenia kompilatorów – najpierw podzespołów, a następnie pełnej Ady. Jedną z pierwszych takich prób było opracowanie kompilatora szerokiego podzespołu Ady dla komputerów SM, w Instytucie Maszyn Matematycznych. Kompilator ten przeszedł badania międzynarodowe w ramach organizacji SM w roku 1985. Kompilator pełnej Ady udało się opracować jedynie na Węgrzech. W połowie 1988 roku był on jeszcze w fazie uruchamiania i wymagał trochę pracy zanim stałby się pełnowartościowym produktem.

OGÓLNA BUDOWA KOMPILATORÓW ADY

Kompilatory Ady zazwyczaj składają się z dwóch części: przedniej (ang. *front end*) i tylnej (ang. *back end*). Część przednia jest częścią niezależną od maszyny i obejmuje wszystkie funkcje, które można zrealizować bez informacji o komputerze docelowym. Część tylna składa się zazwyczaj z analizatora składniowego, analizatora semantycznego i generatora kodu abstrakcyjnego wraz z jego optymalizacją.

Część tylna jest częścią zależną od maszyny, jej podstawowym składnikiem jest generator kodu. W tej części jest dokonywane adresowanie obiektów, są obliczane wyrażenia statyczne z dostępną dla komputera docelowego dokładnością, jest dokonywana optymalizacja.

Bardzo często kompilator Ady składa się z jednej części przedniej i kilku tylnych realizujących generatory kodów dla różnych komputerów docelowych. Kompilator musi też współpracować z biblioteką programową, czyli w praktyce do jego zrealizowania jest konieczne zrealizowanie obsługi bibliotecznej.

Program źródłowy napisany w Adzie jest przekształcany na postać pośrednią w postaci grafu, który jest modyfikowany przez cały czas kompilacji. Postać ta jest najczęściej zapisana w specjalnym języku o nazwie DIANA (ang. *Description Intermediate Attributed Notation for Ada*). DIANA jest językiem pośrednim dla wszystkich faz kompilacji w większości kompilatorów Ady, jest także językiem pośrednim dla przechowywania informacji o jednostkach bibliotecznych.

PROBLEMY REALIZACYJNE

Podstawową trudnością w realizacji kompilatora Ady jest jej złożoność. Pracochłonność opracowania kompilatora Ady bez otoczenia programowego jest obliczana na 55–60 osobolat, natomiast dla porównania pracochłonność kompilatora Pascala wynosi 10 osobolat. Kompilatory Ady są programami dużymi, liczącymi 250 000 do 500 000 wierszy tekstu źródłowego w Adzie (standardowo kompilatory Ady są pisane w Adzie).

Następną podstawową trudnością jest niejednoznaczność normy. Biuro ds. języka Ada wydaje oficjalną interpretację niejednoznaczności raportu. Do chwili obecnej Biuro musiało ustosunkować się do kilku tysięcy pytań, na które nie udaje się odpowiedzieć na gruncie samej normy. Niejednoznaczności te nie mają zazwyczaj znaczenia dla programisty piszącego w Adzie, mają jednak podstawowe znaczenie dla implementatorów.

Trudności związanych z implementacją poszczególnych konstrukcji języka jest bardzo wiele. Niektóre z nich wymagały opracowania nowych algorytmów, nieznanych dotychczas z implementacji innych języków; inne dają się rozwiązać stosunkowo łatwo, ale istotnie zwiększają rozmiar kompilatora. Podstawowe trudności implementacyjne są związane z:

- niejednoznacznością identyfikatorów i operatorów spowodowaną możliwością przeciążeń,
- typami zależnymi od wyróżnika (dla takich typów jest konieczne przekazywanie informacji o typie do programu wynikowego),
- obsługą jednostek bibliotecznych i zapewnieniem właściwego ich powiązania,
- adresowaniem kodu wynikowego, w szczególności przydziałem pamięci dla zadań, a także obiektów niezawężonych,
- dużą liczbą warunków, jakie musi sprawdzić kompilator, zanim uzna konstrukcję za poprawną,
- efektywnością kompilatora i generowanego przez niego kodu,
- specyfiką reprezentacji, która powoduje bardzo duże trudności związane z efektywną realizacją operacji.

Trudności z implementacją Ady rozpoczynają się już na poziomie elementów leksykalnych.

W Adzie zakłada się, że długość identyfiktorów jest ograniczona jedynie przez długość wiersza. Przechowywanie przez cały czas pełnych identyfikatorów jest bardzo kosztowne. Konieczne jest więc ich przekodowywanie np. za pomocą numerów. Przy generowaniu kodu dodatkową trudnością jest fakt, że większość oprogramowania systemowego w istniejących systemach operacyjnych wymaga dla nazw zewnętrznych określonej długości. Nie można więc zastosować nazw pochodzących z programu napisanego w Adzie w kodzie wynikowym.

W Adzie zakłada się też, że zarówno literały, jak i wyrażenia typów uniwersalnych powinny być obliczane dokładnie. Można więc łatwo napisać literał, którego wartość jest poza zakresem komputera, na którym działa kompilator, a literał ten powinien być obliczony dokładnie, np.

$I :=$ 1_000_000_000_000_000 – 999_999_999_999_999;

Następna trudność przy realizacji Ady wynika z faktu, że wiele różnych konstrukcji ma podobną składnię, np. $T(X)$ może zostać zinterpretowane jako:

– wywołanie podprogramu *T* z parametrem *X*,
– kwalifikacja typem,
– element tablicy.

Bardzo duże trudności sprawia interpretacja identyfikatorów i wyrażeń, czego przyczyną jest możliwość przeciążania identyfikatorów. W przykładzie na wydruku 1, w przypadku wywołania funkcji *Wez*, kontekst jest niejednoznaczny i kompilator powinien zasygnalizować błąd. Operacje dodawania różnią się od siebie, a o tym którą z nich należy zastosować decyduje typ zmiennej stojącej po lewej stronie operatora przypisania.

```
declare
   type Kolor is ( Zielony , Zolty, Czerwony );
   type Swiatlo is (Zolty, Czerwony, Zielony);
   function Wez( K : Kolor ) return Kolor;
   function Wez( K : Swiatlo ) return Kolor;
   function "+"( L,P : Swiatlo ) return Swiatlo;
   function "+"( L,P : Kolor ) return Kolor;
   K : Kolor;
   L : Swiatlo;

begin
   K := Wez( Czerwony ); -- co to ma być, kontekst
                         -- niejednoznaczny
   K := Zolty + Czerwony + Zielony; -- typ wyrażenia
           -- jest wyznaczony dopiero przez podstawienie
   L := Zolty + Czerwony + Zielony; -- typ wyrażenia
           -- jest wyznaczony dopiero przez podstawienie
end;
```

Wydruk 1

Zinterpretowanie typu wyrażenia jest dla kompilatora trudne. Udowodniono, że interpretacja wyrażeń wymaga dwukrotnego ich przeglądania w celu określenia wszystkich składowych lub stwierdzenia, że kontekst jest niewystarczający do dokonania interpretacji. Dla kompilatora szczególnie trudne jest to, że kontekst, który należy uwzględnić do zinterpretowania identyfikatorów i operatorów, jest często bardzo szeroki. Przykładem mogą być agregaty rekordowe, lub wywołania nazwane parametrów

procedure *X*(*S* ...; *T* ...; *U* ...);
procedure *X*(*S* ...; *T* ...; *U* ...; *W* ...);

Wywołanie:

X(*U*=⟩ ..., *T*=⟩ ..., *S*=⟩ ..., *W*=⟩ ...);

może być zinterpretowane jako wywołanie procedury czteroparametrowej, dopiero gdy analizowane jest przypisanie do parametru *W*. Dodatkowa kompilacja może nastąpić wtedy, gdy parametr *W* ma wartość domyślną. Może być on wówczas pominięty w wywołaniu i wywołanie takie może być niejednoznaczne.

Kompilator musi sprawdzać poprawność użycia zasobów deklarowanych w innych jednostkach kompilacyjnych, a używany przez jednostkę aktualnie kompilowaną (por. wydruk 2). Kompilator musi także sprawdzać, czy dana kompilacja nie narusza kolejności kompilacji wyznaczonej przez klauzule *with* i pragmy *ELABORATE*.

```
---- Pierwsza jednostka kompilacyjna ------------------

package A is
   type T is .......;
end;

-------------------------------------------------------
---- Druga jednostka kompilacyjna ---------------------

with A;
package B is
   U : A.T;
end;

-------------------------------------------------------
---- Trzecia jednostka kompilacyjna -------------------

with B;
procedure P is
begin
   -- W ciele P zmienna U jest dostępna, ale typ tej
   -- zmiennej jest niedostępny, gdyż P nie dołącza A
end;
```

Wydruk 2

Trudności realizacyjne występują w programach stosujących specyfikację reprezentacji. Klauzule specyfikacji reprezentacji pozwalają, na przykład, na umieszczanie całych rekordów w pojedynczych słowach komputera (wydruk 3 – przykład zaczerpnięty z normy Ady). Klauzula taka nakłada wymagania na adresowanie obiektów typu

```
SLOWO : constant := 4;
type Stan is (A, M, W, P);
type IRYB is (FIX, DEC, EXP, SIGNIF );

type MASKA_BAJTOWA is array (0..8)    of BOOLEAN;
type MASKA_STANU   is array (STAN)    of BOOLEAN;
type MASKA_TRYBU   is array (TRYB)    of BOOLEAN;

type SLOWO_STANU_PROGRAMU is
   record
      MASKA_SYSTEMU      : MASKA_BAJTOWA;
      KLUCZ_DOSTEPU      : Integer range 0..3;
      STAN_MASZYNY       : MASKA_STANU;
      POWOD_PRZERWANIA   : KOD_PRZERWANIA;
      ILC                : Integer range 0..3;
      CC                 : Integer range 0..3;
      MASKA_PROGRAMU     : MASKA_TRYBU;
      ADRESS_WZNOWIENIA  : ADDRESS;
   end record;

for SLOWO_STANU_PROGRAMU use
   record at mod 8;
      MASKA_SYSTEMU      at 0 * WORD range 0 .. 7;
      KLUCZ_DOSTEPU      at 0 * WORD range 10..11;
      STAN_MASZYNY       at 0 * WORD range 12..15;
      POWOD_PRZERWANIA   at 0 * WORD range 16..31;
      ILC                at 1 * WORD range  0.. 1;
      CC                 at 1 * WORD range  2.. 3;
      MASKA_PROGRAMU     at 1 * WORD range  4.. 7;
      ADRESS_WZNOWIENIA  at 1 * WORD range  8..31;
   end record;

for SLOWO_STANU_PROGRAMU use 8 * SYSTEM.STORAGE_UNIT;
```

Wydruk 3

SŁOWO_STANU_PROGRAMU; adres musi być podzielny przez 8, a ponadto wskazuje na jakich pozycjach znajduje się poszczególne pole. Wygenerowanie efektywnego kodu dla dostępu do poszczególnych pól jest bardzo trudne.

Duże trudności napotykają implementatorzy Ady z wymyśleniem struktury programu wynikowego, a przede wszystkim z przydziałem pamięci dla tego programu. Podstawowym problemem jest fakt, że jeśli podprogramy mogą zawierać zadania, to tradycyjny stosowy przydział pamięci dla programu jest niewystarczający. Jednym z pierwszych rozwiązań był stos kaktusowaty. Obecnie istnieje wiele modeli przydziału pamięci, z których najefektywniejszym okazuje się przydział losowy.

Osobnym i bardzo trudnym zagadnieniem jest taka realizacja kompilatora, aby program wynikowy mógł pracować w konfiguracji rozproszonej i to zarówno wieloprocesorowej, jak i z rozproszeniem pamięci. Powyższe przykłady ilustrują zaledwie kilka z bardzo wielu kłopotów, z którymi borykają się twórcy kompilatorów Ady. Kłopoty te zostają jednak przezwyciężane i obecnie istnieje wiele legalizowanych kompilatorów.

LEGALIZACJA

W celu zapewnienia przenośności programów Ady opracowane zostały środki legalizacji kompilatorów Ady (ACVC – *Ada Compiler Validation Capability*). Środki te obejmują Podręcznik Implementatorów Ady, w którym opisano zalecenia, jak należy implementować poszczególne konstrukcje języka, i procedurę legalizacyjną, która obejmuje około dwóch tysięcy stosunkowo krótkich testów. Testy są podzielone na następujące grupy:

A. Poprawne programy Ady; kompilator nie powinien sygnalizować żadnych błędów.

B. Programy niepoprawne; kompilator musi wykryć wszystkie konstrukcje błędne i jednocześnie nie może sygnalizować błędów przy konstrukcjach poprawnych.

C. Programy poprawne przeznaczone do wykonywania: ocena kompilatora wypada pomyślnie, jeżeli testy dają poprawne wyniki.

D. Programy sprawdzające sposób implementacji; brak jasnych kryteriów, kiedy kompilator jest uznawany za poprawny.

E. Testy niejednoznaczności raportu sprawdzające, w jaki sposób niejednoznaczności raportu zostały zaimplementowane w danym kompilatorze.

Obecnie Ada staje się normą ISO, a w końcu 1987 roku przestała być znakiem handlowym Departamentu Obrony USA. W związku z tym procedura legalizacyjna jest wymagana jedynie dla kompilatorów używanych do prac wykonywanych na zlecenie Armii USA.

Początkowe próby implementacji Ady dawały kompilatory, które działały z szybkością około 1–2 wierszy tekstu źródłowego na sekundę i generowały bardzo mało efektywny kod. Obecnie sytuacja zmieniła się radykalnie. Kompilatory działają stosunkowo szybko, a kod przez nie generowany jest porównywalny pod względem efektywności z kodem generowanym przez najlepsze kompilatory języka C.

INTERSOFT

00–478 Warszawa, Al. Ujazdowskie 18 m. 14
Tel. 28-01-76

Szanowni Państwo!

Polecamy duży wybór dokumentacji dot. komputerów IBM oraz oprogramowania, m.in.:

Przewodnik programisty IBM	144 000 zł	Turbo Power Tools Plus (procedury do TP 4/5)	115 000 zł
Wprowadzenie do komputerów IBM	32 000 zł	Grafika TP v.4.0 i TC v.1.5	110 000 zł
Sidekick	50 000 zł	Clipper 87, kompil. do dBase III+	160 000 zł
Wstęp do grafiki (Basic, Turbo, Graphics)	60 000 zł	dBase III+	170 000 zł
Autocad v.2.17	86 000 zł	dBase III+, praca w sieci	40 000 zł
Lotus 1–2–3 v.2.0	100 000 zł	Fox Base+	140 000 zł
Framework IIp	114 000 zł	Programowanie w assemblerze	160 000 zł
System operacyjny DOS 4.0	160 000 zł	Eureka	65 000 zł
System operacyjny DOS 3.3	144 000 zł	Poly-Windows	48 000 zł
Turbo Basic	130 000 zł	Instrukcja obsługi PC 1512	150 000 zł
Turbo „C" v.2.0	145 000 zł	Wordstar 2000	50 000 zł
Aztec „C"	132 000 zł	Modula 2 Logitech	102 000 zł
Zastosowanie języka C dla zaawansowanych	160 000 zł	Informix	180 000 zł
Turbo Graphics (do TP v.3.0)	74 000 zł	OS-2, opis systemu	130 000 zł
Turbo Pascal v.4.0	43 000 zł	Or-Cad	170 000 zł
Turbo Pascal v.5.0	270 000 zł	Novell, podr. użytkownika i instalatora	180 000 zł
Turbo Pascal v.5.5	306 000 zł		

Do większości pozycji dołączamy dyskietki z przykładami (koszt nośnika nie jest ujęty w wyżej wymienionych cenach).

Prowadzimy sprzedaż wszelkiego sprzętu komputerowego po bardzo konkurencyjnych cenach.

Do sprzedawanego przez nas sprzętu dodajemy bezpłatnie pakiety dokumentacji i oprogramowania o wartości zależnej od wielkości sprzedaży.

ZAPRASZAMY!

EO/232/89

Pierwsza wersja artykułu została przedstawiona podczas konferencji PTI „Współczesne obszary zastosowań informatyki", Szczyrk, 3–5 maja 1989 r.

JANUSZ ZALEWSKI
Instytut Energii Atomowej
Otwock-Świerk

Okres istnienia oprogramowania (1)

Coraz szersze stosowanie komputerów w różnych dziedzinach gospodarki spowodowało gwałtowny wzrost zainteresowania jakością oprogramowania – szczególnie w tzw. **zastosowaniach krytycznych**, tj. takich, *które stanowią zagrożenie dla życia lub zdrowia, ewentualnie mogą spowodować znaczne straty materialne*. Wkrótce też zaczęto określać cechy, jakie powinno mieć oprogramowanie, aby można mówić o jego dobrej jakości. Okazało się wszakże, że znacznie łatwiej te cechy wyliczyć niż je bezpośrednio lub pośrednio zmierzyć, co dało początek całej nowej dziedzinie wiedzy informatycznej, tzw. programometrii. Od opublikowania pierwszych prac na temat różnych aspektów mierzenia jakości oprogramowania minęło już kilkanaście lat. Choć ta ważna dziedzina stanowi punkt wyjścia do rozważań nt. jakości oprogramowania, nie będę się nią zajmował, odsyłając czytelników do innych opracowań, np. [2], pełne jej omówienie wymagałoby bowiem oddzielnego wykładu. Zainteresowanym przedmiotem mogę polecić najpełniejszy, jaki znam, spis literatury [10], którego pewną liczbę egzemplarzy może rozesłać redakcja – aż do wyczerpania nakładu. W poniższym rozdziale przedstawię tylko sposób podejścia do tego problemu.

WSPÓŁCZESNE POGLĄDY NA JAKOŚĆ OPROGRAMOWANIA

Sądzę, że ciekawe byłoby przedstawienie przynajmniej niektórych cech mających wpływ na jakość oprogramowania. W nowym wydaniu słownika inżynierii oprogramowania IEEE [4] wyliczono około 30 takich cech: dyspozycyjność, efektywność, elastyczność, integralność, konserwowalność, krytyczność, modularność, nadmiarowość, niesprzeczność, niezawodność, odporność, ogólność, poprawność, przenośność, prostota, realizowalność (ang. *feasibility*), rozszerzalność, spoistość (ang. *cohesion*), sprzęgliwość (ang. *coupling*), śladowalność (ang. *treaceability*), testowalność, tolerowanie błędów i defektów, użyteczność, wielużywalność (ang. *reusability*), współoddziaływanie (ang. *interoperability*), wydajność, zabezpieczenie (ang. *security*), zgodność, złożoność. Zacytowanie ich pełnych definicji zajęłoby zbyt dużo miejsca, dlatego przeniesiono je do rubryki terminologicznej następnego numeru **INFORMATYKI.**

Inna organizacja mająca szczególny tytuł do formułowania zaleceń dotyczących jakości oprogramowania w zastosowaniach krytycznych, Międzynarodowa Agencja Energii Atomowej, wyliczyła również około 30 cech charakterystycznych, świadczących o jego jakości [3], bardzo zbliżonych do podanych przez IEEE (pełne definicje znajdą się również w następnym numerze): czytelność, dokładność, efektywność, ergonomiczność, integralność, komunikatywność, konserwowalność, modyfikowalność, niesprzeczność, niezależność od urządzeń, niezawodność, odporność, poprawność, prostota, przenośność, rozszerzalność, samoopisywalność (ang. *self-descriptiveness*), samozwartość (ang. *self-containedness*), sondowalność (ang. *accountability*), strukturalność, śladowalność, testowalność, użyteczność, wielużywalność, współdziałalność, zdolność obsługi błędów, zrozumiałość, zupełność, zwartość.

Oczywiście, cechy te w większości, może nawet wszystkie, są niemierzalne, tzn. niewyrażalne w postaci liczbowej. Aby można je określić ilościowo, trzeba skonstruować specjalne miary, tzw. metryki. Według słownika [4] **metryką** nazywa się *ilościową miarę stopnia, w jakim system, składowa lub proces posiada określony atrybut* (ang. *a quantitative measure of the degree to which a system, component or process possesses a given attribute*). Przez atrybut rozumie się, jak w języku potocznym, pewną cechę, definiując ten termin ogólnie jako charakterystykę obiektu (ang. *a characteristic of an item*), a dokładniej, definiując **atrybut jakości** jako cechę lub charakterystykę wpływającą na jakość obiektu (ang. *a feature or characteristic that affects item's quality*).

Choć znalezienie takich metryk jest trudne, nie jest przecież niemożliwe. Jako przykład można podać bardzo interesujące skonstruowanie dwóch metryk mogących służyć za miary dwóch atrybutów z listy IAEA, tj. modyfikowalności i strukturalności, dla programów napisanych w języku Ada [1]. **Modyfikowalność** oprogramowania, czyli łatwość wprowadzania zmian, można powiązać bezpośrednio z **odpornością na zmiany**, wyrażoną jako *stosunek liczby odwołań do składowych obiektów złożonych o nielokalnych typach danych, do liczby wszystkich wierszy programu*. Znaczenie tej metryki w Adzie odpowiada znanemu faktowi, że rozproszenie informacji utrudnia, a skoncentrowanie jej w prywatnych częściach pakietów ułatwia modyfikowanie programu. **Strukturalność** można z kolei mierzyć za pomocą stopnia odseparowania poszczególnych pakietów, używając do tego cechy zwanej w Adzie **widocznością pakietu.** Przykładowo, jedną z metryk można zdefiniować jako *stosunek liczby jednostek programowych, w których informacja z danego pakietu jest używana, do liczby jednostek, w których pakiet może być uczyniony widocznym przez użycie klauzuli* **with.** Znaczy to, że im mniejsza będzie wartość tej metryki, tym większe będzie odseparowanie modułów i lepsza strukturalność programu. W cytowanym artykule [1] podano też inną metrykę widoczności, co świadczy o trudnościach we właściwym doborze tych miar.

W jednym z nowszych opracowań Komitetu Technicznego nr 7 EWICS [11], w części dotyczącej klasyfikacji atrybutów i miar jakości oprogramowania, podano przykłady takiej dekompozycji atrybutów jakości, aby można je mierzyć ilościowo. Przykładowo, **użyteczność** rozumiana jako *łączny wysiłek wymagany do nauczenia się systemu, eksploatowania go, przygotowywania danych i interpretowania wyników* [3], tj. zasób umiejętności koniecznych do użytkowania systemu, proponuje się określić za pomocą kombinacji czterech cech niższego poziomu, takich jak:

wymagania podstawowe (ang. *entry requirement*) opisujące minimalne umiejętności użytkownika, np. fizyczne, jak wzrok, słuch, lub mentalne, jak inteligencja, wykształcenie, doświadczenie itp.,

konieczność uczenia się (ang. *learning requirement*), mierzona przede wszystkim czasem potrzebnym do opanowania i kontroli głównych czynności systemu,

zdolność usługowa (ang. *handling ability*) mierzona czystą produktywnością w ciągu ustalonego czasu,

upodobanie (ang. *likability*) określające, w jakim stopniu użytkownicy lubią korzystać z systemu, co można zmierzyć np. częstością zmian miejsca przez personel lub ankietując pracowników.

Inny ważny atrybut oprogramowania – **konserwowalność** – rozłożono jeszcze bardziej szczegółowo na łatwiej mierzalne cechy, wyrażalne bezpośrednio w jednostkach czasu, tzn.:

- czas rozpoznania defektu (automatycznie lub przez człowieka),
- opóźnienie administracyjne (np. do chwili uaktywnienia mudułu programowego, lokalizującego defekt),
- czas skompletowania narzędzi (np. dokumentacji analizatorów programowych, zbioru testów, wyników wzorcowych itp.),
- czas analizy defektu, tj. prześledzenia pracy wstecz, od objawów defektu do jego źródeł,
- czas formułowania poprawek, tj. czas potrzebny na przełożenie wniosków o przyczynie defektu na propozycję podjęcia odpowiednich czynności,
- czas inspekcji poprawek, rozumiany jako czas niezbędny do sprawdzenia hipotezy, np. ze względu na zupełność i niesprzeczność,
- czas wprowadzania poprawek,
- czas testowania,
- czas oceny wyników testu,
- czas przywrócenia poprawnej pracy.

Oczywiście, nie wszystkie te cechy muszą wchodzić w skład miary konserwowalności, ponieważ nie wszystkie zależą od samego oprogramowania – część z nich zależy w dużym stopniu od otoczenia.

Jak widać, bezpośrednie i pośrednie wyrażanie jakości oprogramowania w sposób ilościowy nie jest łatwe. Dlatego dobrą jakość produktów

próbuje się zapewnić głównie drogą narzucenia pewnych rygorów na proces wytwarzania oprogramowania. Z dzisiejszej perspektywy są widoczne w zasadzie dwie drogi prowadzące do tego celu.

Ponieważ do niedawna przez programowanie rozumiano wyłącznie kodowanie, pierwsze wysiłki w kierunku polepszenia jakości oprogramowania skierowano w stronę języków programowania. Ta tendencja doprowadziła niewątpliwie do rozwoju języków, tj. udoskonalenia ich konstrukcji w celu polepszenia jakości programów, czego wyrazem są niewątpliwie takie języki, jak Ada czy Modula 2, oraz – co równie ważne – do unowocześnienia metod (szczególnie metod formalnych) i narzędzi programowania (włącznie z tworzeniem tzw. środowisk programistycznych). Ten obszar wiedzy informatycznej jest dziś bardzo rozległy – o czym świadczy liczba stosowanych i wciąż powstających nowych języków, metod i narzędzi, i nie będzie tu omawiany.

Równocześnie jednak zaczęto zdawać sobie sprawę, że procesu programowania nie należy ograniczać do kodowania, lecz można i trzeba utożsamiać go z wytwarzaniem produktu programowego, w sposób analogiczny do technologii wytwarzania wyrobów w innych dziedzinach przemysłu – z podziałem na poszczególne etapy, fazy lub operacje. Podział procesu produkcji oprogramowania na szereg wyodrębnionych i dobrze rozróżnialnych faz, ze ściśle określonymi przejściami, ma służyć właśnie ulepszeniu końcowego produktu pod względem jakości, wskutek właściwego przestrzegania zasad produkcyjnych. Choć możliwych i stosowanych jest wiele modeli tzw. cyklu produkcyjnego oprogramowania (ang. *software development cycle*), szczególną popularność zdobywają zasady wypracowane przez komitet techniczny ds. inżynierii oprogramowania Towarzystwa Komputerowego amerykańskiej organizacji inżynierów elektryków i elektroników (*IEEE Computer Society, Technical Committee on Software Engineering*), ogłoszone w większości w postaci różnego rodzaju dokumentów normalizacyjnych.

OKRES ISTNIENIA OPROGRAMOWANIA WEDŁUG NORM IEEE

Zbiór norm amerykańskich dotyczących inżynierii oprogramowania, opracowanych przez *IEEE Computer Society, Technical Committee on Software Engineering*, składa się z kilkunastu dokumentów.

Według definicji podanej w [4] **okres istnienia oprogramowania** (ang. *software life cycle*) jest *to odcinek czasu od powzięcia decyzji o wytworzeniu produktu programowego do momentu, gdy ten produkt przestanie być dostępny*. Jest oczywiste, że tak długi odcinek czasu musi dzielić się na krótsze tzw. fazy. Podział całego okresu istnienia na fazy może wyglądać bardzo różnie – zależy to od rodzaju produktu. Przykładowo, w tzw. **modelu kaskadowym** (ang. *waterfall model*), według normy [4] może on dzielić się na analizę wymagań, projektowanie wstępne i szczegółowe, kodowanie, integrowanie modułów, testowanie, dokumentowanie, dostawę, eksploatację, konserwowanie i wycofanie. W bardziej szczegółowym podziale, przygotowywanym w postaci normy [9], dzieli się cały okres istnienia na procesy, wyróżniając kilka grup, a mianowicie procesy:
- zarządzania przedsięwzięciem
- przedprodukcyjne
- produkcyjne
- poprodukcyjne
- integracyjne.

Procesy zarządzania i procesy integracyjne trwają przez całe przedsięwzięcie, natomiast właściwe procesy produkcyjne następują po sobie kolejno w czasie. Każdy z procesów jest realizowany przez wykonanie wielu ściśle określonych czynności (ang. *activities*).

Grupa **zarządzania przedsięwzięciem** składa się z trzech procesów: zainicjowania przedsięwzięcia (ang. *project initiation*), nadzorowania i sterowania przedsięwzięciem (ang. *project monitoring and control*), zarządzania jakością oprogramowania (ang. *software quality management process*). Przykładowo, do czynności wykonywanych w tym ostatnim procesie zalicza się kolejno: planowanie zapewnienia jakości oprogramowania, opracowanie miar jakości, zarządzanie kontrolą jakości i określenie potrzeb polepszenia jakości.

Procesy przedprodukcyjne podzielono na dwa rodzaje: opracowanie koncepcji (ang. *concept exploration*) i rozdział pracy (ang. *system allocation process*). Na grupę procesów bezpośrednio **produkcyjnych** składają się procesy: specyfikowania wymagań (ang. *requirements specification*), projektowania (ang. *design*) i implementowania (ang. *implementation*). **Procesy poprodukcyjne** obejmują: instalowanie, eksploatację i serwis, wycofanie z użycia i konserwowanie.

Grupa procesów integracyjnych obejmuje cztery procesy: weryfikację i atestację, zarządzanie konfiguracją, wytwarzanie dokumentacji oraz szkolenie. Poniżej omówię przykładowe czynności niektórych procesów lub przykładowe procesy każdej z grup.

PRZYKŁAD CZYNNOŚCI PROCESU ZARZĄDZANIA PRZEDSIĘWZIĘCIEM

Jednym z trzech procesów służących zarządzaniu przedsięwzięciem (ang. *project management*), zidentyfikowanym w normie [9], jest proces kontroli jakości. Jest on wykonywany przez cały okres istnienia oprogramowania – nie tylko podczas jego wytwarzania. Jedna z pierwszych norm kontroli jakości oprogramowania, norma amerykańska [5], dotyczyła zasadniczej czynności – zapewnienia jakości. Obecnie, zmierzają ku końcowi prace nad aktualizacją tej normy. Jej zalecenia dotyczą przede wszystkim tzw. **oprogramowania krytycznego,** przez co rozumie się *oprogramowanie, którego awarie mogą zagrozić bezpieczeństwu lub spowodować znaczne straty materialne lub społeczne* (ang. *software whose failures would impact safety or cause large financial or social losses*).

Plan kontroli jakości oprogramowania

Zgodnie z tą normą, dokument zwany **planem zapewnienia jakości oprogramowania** (ang. *software quality assurance plan*) powinien obejmować piętnaście rozdziałów w następującej kolejności, według tytułów:

1. Cel (określający cel i zakres planu dla poszczególnych modułów oraz ich przeznaczenie);
2. Wykaz dokumentów (na które powołano się w planie);
3. Zarządzanie (rozumiane jako struktura organizacyjna, podział zadań i podział odpowiedzialności za ich realizację);
4. Dokumentacja (obejmująca co najmniej minimalny zbiór dokumentów niezbędnych do zapewnienia jakości, jak specyfikacja wymagań, opis projektu, plan weryfikacji i atestacji, sprawozdanie z weryfikacji i atestacji, dokumentacja użytkowa, plan zarządzania konfiguracją);
5. Normy, praktyki, konwencje i metryki (stosowane do zapewnienia jakości oprogramowania – dokładnie chodzi o standardy dokumentowania, określania struktury logicznej, kodowania, komentowania, wykonywania przeglądów i lustracji oraz testowania);
6. Przeglądy i lustracje (ang. *reviews and audits*), które powinny obejmować co najmniej następujące czynności: przegląd wymagań (ang. *software requirements review*, SRR), wstępny przegląd projektu (ang. *preliminary design review*, PDR), krytyczny przegląd projektu (ang. *critical design review*, CDR), przegląd planu werfikacji i atestacji (ang. *software verification and validation plan review*, SVVPR), lustracja funkcjonalna (ang. *functional audit*), lustracja fizyczna (ang. *physical audit*), lustracja produkcyjna (ang. *in-process audit*), przeglądy zewnętrzne (ang. *managerial reviews*), przegląd planu zarządzania konfiguracją (ang. *software configuration management plan review*, SCMPR), przegląd końcowy (ang. *post-mortem review*);
7. Testy (dodatkowe, tzn. nie objęte planem werfikacji i atestacji);
8. Zaistniałe problemy i działania korekcyjne (określające zasady postępowania w wypadku wykrycia defektów);
9. Narzędzia, techniki i metodyki (stosowane z myślą o zapewnieniu jakości oprogramowania);
10. Kontrola kodu (przed nieupoważnionym użyciem podczas całego okresu istnienia oprogramowania);
11. Kontrola nośników (przed nieupoważnionym dostępem);
12. Kontrola dostawców;
13. Gromadzenie, utrzymywanie i przechowywanie dokumentów (służących zapewnieniu jakości oprogramowania);
14. Szkolenie (konieczne dla zapewnienia jakości oprogramowania);
15. Analiza ryzyka.

Ważną częścią tej normy jest zidentyfikowanie pięciu metryk, opartych na normach [6, 7], związanych z jakością oprogramowania, są to metryki:
- rozgałęzień,
- punktów decyzyjnych,
- dziedzinowa,
- komunikatów błędów,
- wykonania wymagań.

Przykładowo, przez **metrykę rozgałęzień** (ang. *branch metric*) rozumie się *stosunek liczby modułów, w których każde rozgałęzienie zostało*

wykonane przynajmniej raz, do całkowitej liczby modułów. W tym samym duchu są utrzymane pozostałe definicje, np. **metryka dziedzinowa** (ang. *domain metric*) jest *to stosunek liczby modułów, w których poprawnie przetworzono przynajmniej jeden zestaw ważnych i jeden zestaw nieważnych danych wejściowych każdej klasy (np. komunikatów zewnętrznych, poleceń operatorskich, danych lokalnych itp.), do całkowitej liczby modułów.*

ZALECENIA DOTYCZĄCE STOSOWANIA PLANU

Ważnym dokumentem uzupełniającym tę normę są zalecenia dotyczące jej użycia [8]. Każdy z rozdziałów planu zapewnienia jakości jest tu omówiony dokładniej wraz z dodatkowymi wyjaśnieniami. Przykładowo, dokumentacja punktu 4. może być rozszerzona o plan wytwarzania oprogramowania (ang. *software development plan*), podręcznik norm i zasad (ang. *standards and procedures manual*), plan zarządzania przedsięwzięciem (ang. *software project management plan*) i podręcznik konserwacji oprogramowania (ang. *software maintenance manual*), a także dodatkowe dokumenty, jak wymagania użytkowe, specyfikacja sprzężeń wewnętrznych i zewnętrznych, instrukcja eksploatacji i instrukcja instalowania, podręcznik i plan szkolenia. Zalecana zawartość każdego z tych dokumentów jest oczywiście omówiona mniej lub bardziej szczegółowo, zależnie od jego wagi dla zapewnienia jakości. W punkcie 5. z kolei wyjaśniono rozróżnienie **konwencji,** jako standardowych wymagań dotyczących produktu programowego, od **praktyk,** rozumianych jako standardowe wymagania narzucone na proces wytwórczy oprogramowania. Praktyki i konwencje uznane za konieczne do stosowania objęte są normami.

Najobszerniejsze jednak, bo najważniejsze dla zapewnienia jakości, są zawarte w tym rozdziale wyjaśnienia dotyczące punktu 6. planu – przeglądy i lustracje. Każdy element oprogramowania i każdy dokument ujęty w procesie wytwórczym, służący zapewnieniu jakości oprogramowania, musi przejść przez wieloraką kontrolę. Szczególnie ważne są dwa rodzaje tej kontroli:

- **przeglądy,** które polegają na badaniu integralności projektu, wykrywaniu usterek i wprowadzaniu niezbędnych zmian, mają więc znaczenie techniczne
- **lustracje,** które stanowią podstawę do podejmowania decyzji, a więc mają znaczenie dla szczebli kierowniczych.

Dla wszystkich rodzajów przeglądów i lustracji opisano ich przebieg, podając mniej lub bardziej szczegółowe zasady postępowania, np. zalecając podczas przeglądu specyfikacji wymagań sprawdzenie jej m.in. ze względu na następujące cechy:

- śladowalność i zupełność,
- stosowność wymagań, algorytmów i równań,
- poprawność opisów logicznych,
- zgodność sprzężeń zewnętrznych,
- dostosowanie opisów do sprzęgu człowiek-maszyna,
- niesprzeczność użytych symboli i specyfikacji sprzężeń,
- dostępność stałych i tabel do wykonywania obliczeń,
- testowalność wymagań,
- stosowność i zupełność wymagań werfikacji i odbioru.

Spośród wszystkich rodzajów przeglądów, jedynie przegląd zewnętrzny musi być powierzony niezależnej jednostce organizacyjnej lub oddzielnemu kontrahentowi. Trzy rodzaje lustracji różnią się natomiast celami i momentami ich przeprowadzenia. **Lustracja funkcjonalna** jest wykonywana przed dostawą oprogramowania i ma na celu sprawdzenie, czy spełnione zostały wszystkie wymagania zawarte w specyfikacji. **Lustracja fizyczna** jest przeprowadzana w celu sprawdzenia, że oprogramowanie i jego dokumentacja są wewnętrznie niesprzeczne i gotowe do dostawy. **Lustracja produkcyjna** polega na wykonaniu kontroli implementacji, tj. zgodności kodu z dokumentacją, projektem i wymaganiami, w celu oceny postępów procesu produkcyjnego.

Oprócz szczegółowych wyjaśnień dotyczących zawartości planu zapewnienia jakości oprogramowania, w trzech odrębnych rozdziałach zaleceń [8] opisano też zasady realizacji tego planu, jego oceny oraz modyfikowania.

STUDIUM REALIZOWALNOŚCI – PRZYKŁAD CZYNNOŚCI PROCESU PRZEDPRODUKCYJNEGO

W pierwszym z procesów przedprodukcyjnych, w normie [9] zidentyfikowano pięć czynności:

- określenie warunków i potrzeb,
- sformułowanie potencjalnych rozwiązań,
- przeprowadzenie studium realizowalności,
- zaplanowanie przeniesienia (jeżeli ma zastosowanie),
- sprecyzowanie i sfinalizowanie warunków i potrzeb.

Choć dla żadnej z tych czynności nie opracowano szczegółowej normy, jak na przykład dla zapewnienia jakości, można przytoczyć interesującą propozycję jednej z ciekawszych i chyba nieodzownych czynności, jaką jest studium realizowalności (ang. *feasibility study*). Zaproponowano, aby punktem wyjścia do przeprowadzenia studium realizowalności były wyniki innych czynności tego procesu, a więc:

- wstępne określenie potrzeb,
- zidentyfikowanie możliwych rozwiązań,
- przewidywane ograniczenia i korzyści.

W prowadzeniu studium zaleca się, co jest naturalne, intensywne korzystanie z technik modelowania i makietowania (ang. *prototyping*). Produktem tej czynności powinny być zalecenia dotyczące wykonania planowanego przedsięwzięcia, np. określające, czy jest możliwe jego wykonanie, a jeśli tak, to czy oprogramowanie należy wykonać, czy zakupić itd. Z nielicznej literatury dotyczącej tej czynności zwraca uwagę raport [12], ponieważ precyzuje pewne wymagania normalizacyjne narzucone na przeprowadzenie studium realizowalności. Tematowi temu warto poświęcić trochę uwagi także z tego powodu, że jak sądzę, nie stykamy się z tego rodzaju problemami w krajowej praktyce programistycznej.

Studium realizowalności może być wykonywane zarówno wewnątrz organizacji planującej zastosowanie lub wytworzenie oprogramowania, jak i na zewnątrz niej, tj. przez oddzielnego kontrahenta. Jednakże, zlecenie przeprowadzenia takiego studium wykonawcy zewnętrznemu jest bardziej celowe o tyle, że wtedy łatwiej ustalić, czego oczekuje się jako wyniku studium. Punktem wyjścia powinno więc być jednoznaczne stwierdzenie, na jakie pytanie lub pytania zleceniodawca oczekuje odpowiedzi, tj. sformułowanie problemu.

W zasadzie studium realizowalności należy wykonać przed rozpoczęciem lub na samym początku przedsięwzięcia programistycznego. Skutkiem przeprowadzenia takiego studium powinno być jednoznaczne stwierdzenie lub wskazanie, czy należy podjąć się realizacji danego przedsięwzięcia, a w wypadku pozytywnej odpowiedzi – także wskazówki dotyczące jego ewentualnego przebiegu. Przy podejmowaniu rozstrzygającej decyzji należy uwzględnić główne kryteria, takie jak maksymalne nakłady, horyzont czasowy, liczba wykonawców, kwalifikacje personelu, możliwości technologiczne, warunki środowiskowe itp.

Ponieważ studium polega w zasadzie na wykonaniu trzech zespołów operacji, tj. zebraniu informacji, przeanalizowaniu jej i opracowaniu wniosków, końcowy dokument, zwany **sprawozdaniem z badania realizowalności** (ang. *feasibility study report*, FSR), powinien zawierać co najmniej:

- wyraźne sformułowanie problemu (pytania, na które przeprowadzane studium ma odpowiedzieć),
- określenie ograniczeń nałożonych na studium, takich jak termin realizacji, dostęp do informacji itp., a więc także na wiarygodność wyników studium,
- zalecenia dotyczące rodzaju informacji, na których oparto studium,
- różne opcje rozwiązania problemu,
- sugestie dotyczące sposobu i czasu implementacji poszczególnych opcji,
- oszacowanie kosztów poszczególnych opcji rozwiązania, zarówno w procesie produkcyjnym, jak i związanych z późniejszą konserwacją,
- wybór najkorzystniejszej opcji rozwiązania według jasno sformułowanego kryterium.

Warto podkreślić, że oprócz fachowości wykonawców istotny wpływ na wiarygodność wyników studium ma czas jego realizacji (tzn. czy nie jest za krótki) oraz dostęp do informacji (tzn. czy jest pełny). Często zdarza się bowiem, że przyszły użytkownik, wytwórca czy zleceniodawca, w chwili wykonywania studium realizowalności nie jest jeszcze w pełni świadomy całego zakresu przedsięwzięcia programistycznego i wszystkich cech oprogramowania – dlatego nie jest w stanie udzielić wystarczających informacji; może to podważyć wiarygodność wyników studium.

Na tej podstawie, autorzy omawianego dokumentu [12] zawarli w nim zalecenia dotyczące treści, złożonego z dziewięciu rozdziałów, sprawozdania z badania realizowalności.

dokończenie na s. 23

SPIS TREŚCI ROCZNIKA
INFORMATYKA 1989

ARTYKUŁY PROBLEMOWE

	nr	s.
AL-ASSAF KHIDER – Wybrane aspekty bezpieczeństwa danych w informatycznych systemach rachunkowości bankowej	2	21
BIELECKI JAN – Grafika w Turbo Pascalu 4.0	6	15
BIELECKI JAN – Turbo C		
Wywoływanie funkcji, operacje na rejestrach, wstawki asemblerowe	1	26
Jawne i niejawne konwersje danych	3	21
Programowanie przenośne	4	24
BIEŃKO ZYGMUNT – Kryzys polskiej informatyki	2	14
BILSKI EUGENIUSZ – Wrocławskie Zakłady Elektroniczne ELWRO. Okres maszyn cyfrowych typu ODRA	8–12	26
BLIKLE ANDRZEJ – O uwarunkowaniach rozwoju systemów informatycznych i przemysłu oprogramowania	2	4
CHYBA ANDRZEJ – Charakterystyka stanu rozwoju informatyki w kolejowym transporcie wewnątrzzakładowym	2	24
DEMINET JAROSŁAW – Badanie przepustowości sieci lokalnych	5	8
FIETT JERZY – Problemy realizacji technicznej polskich komputerów do 1968 roku	8–12	6
FUGLEWICZ JACEK, KORNIAK TADEUSZ – Oprogramowanie baz danych dla systemów mikrokomputerowych rodziny Mera 600	5	15
GIZDOŃ HENRYK, PAWLAK ADAM, WRONA WŁODZIMIERZ – Zastosowanie języka VHDL do opisu i weryfikacji projektów układów cyfrowych	3	14
GIZDOŃ HENRYK, PAWLAK ADAM, WRONA WŁODZIMIERZ – Język opisu VHDL – podstawowe mechanizmy		
Część 2	1	23
GŁĘBICKI JAROSŁAW – System dBase III Plus w sieciach lokalnych	3	8
GONDZIO MAREK, PAWŁOWSKI MAREK, WYTRĘBOWICZ JACEK – Uruchamianie urządzeń mikroprogramowanych		
Część 1	7	6
GRENIEWSKI MAREK J. – Kilka uwag o powołaniu Centrum Obliczeniowego PAN	8–12	25
HORNOWSKI MICHAŁ – Metoda instalowania bardzo dużych programów	4	20
ISZKOWSKI WACŁAW – Przyspieszanie obliczeń przez ich rozpraszanie		
Część 1. Rozpraszanie sprzętowe	5	1
Część 2. Rozpraszanie programu	6	9
Część 3. Podsumowanie	7	15
KACZMARCZYK KAZIMIERZ – Charakterystyka modułu procesora mikrokomputerów rodziny Mera 600	5	11
KANDZIORA-BLAUMAN KRYSTYNA, KISILEWICZ JERZY – Mikroprocesorowa realizacja protokołu liniowego w systemie SM	5	25
KĄCKI EDWARD, KIERZKOWSKI ZBIGNIEW – Wnioski V Krajowej Konferencji Informatyków, Poznań, 8–10. XII. 1987. Rozwój metod i zastosowań informatyki	2	9
KILIŃSKI ANTONI – O osiągnięciach Instytutu Informatyki Politechniki Warszawskiej zastosowanych w praktyce	8–12	21
KLEPACZ WŁADYSŁAW – 40 lat informatyki w Polsce	8–12	1
KŁOPOTEK MIECZYSŁAW A. – TMK – pakiet syntezy mowy dla IBM PC	1	10
KOCZKODAJ WALDEMAR W. – Arkusz kalkulacyjny jako kolejny krok w kierunku nieprogramowania	6	1
KOZIARSKI KAZIMIERZ – Technologia konwersji oprogramowania cobolowego na IBM PC	4	6
KURAŚ JACEK, LEMBAS JACEK, SKOMOROWSKI MAREK – Język symulacji systemów o działaniu ciągłym ze zdarzeniami dyskretnymi	7	9
LAMBORGINI BRUNO – Wpływ informacji i technologii informacyjnej na strukturę firmy	2	17
LAMPORT LESLIE – Uproszczone specyfikowanie systemów współbieżnych		
Część 3	1	2
LUBIŃSKA TERESA – Makietowanie – ewolucyjne projektowanie systemu informacyjnego do wspomagania procesów decyzyjnych	2	28
ŁUKASZEWICZ LEON – O początkach informatyki w Polsce. Od Grupy Aparatów do Instytutu Maszyn Matematycznych	8–12	2
MACHURA MAREK – Pro domo sua	1	1
MARCZYŃSKI ROMUALD W. – Jak budowałem aparaty matematyczne w latach 1948–1958	8–12	16
MOSZYŃSKI KRZYSZTOF – Moja praca w Biurze Obliczeń i Programów oraz w Zakładzie Aparatów Matematycznych PAN	8–12	13
MAZURKIEWICZ ANTONI – Początki programowania w Polsce. Jak się programowało XYZ	8–12	10
NOWIŃSKI WIESŁAW L. – Rekonstruowanie obrazów w tomografii komputerowej	3	18
NOWIŃSKI WIESŁAW L. – Algorytmy rekonstruowania obrazów za pomocą metod transformacyjnych	4	10
NOWIŃSKI WIESŁAW L. – Algorytmy rekonstruowania obrazów metodami rozwinięcia w szereg	5	21
OSSOWSKA STANISŁAWA – Dobór metody projektowania systemów informatycznych zarządzania	4	22
PAWLAK BOGUMIŁA – MIC – metaasembler mikroprogramów	1	15
PAWŁOWSKI MAREK – Przegląd układów szybkiego mnożenia	4	1
PIETROŃ ROMAN, RZOŃCA WALDEMAR – Symulatory organizacji gospodarczych na Merze 400	2	27
PORĘBSKI ZDZISŁAW – Osobiste refleksje z V Krajowej Konferencji Informatyków	2	11
RYBNIK JANUSZ, SOLAK JERZY – Stacje robocze SUN-3		
Część 1. Architektura	3	4
Część 2. Oprogramowanie	4	16
REDMILL FELIX – Najnowsze zalecenia dotyczące systemów komputerowych	7	1
SIKORSKI ANDRZEJ – Lokalne sieci światłowodowe	3	24
SIMON HERBERT A. – Procesy przeszukiwania i wnioskowania w rozwiązywaniu problemów		
Część 1	3	1
Część 2	4	12
Część 3	5	5
SKRZYMOWSKI JACEK – Nowa norma Fortranu		
Część 1	5	18
Część 2	6	12
SZCZERBIŃSKI ZDZISŁAW – Nowa generacja superkomputerów Cray		
Część 1. Cray X-MP	6	6
Część 2. Cray-2	7	19
STRUK ZBIGNIEW, ZABŻA-TARKA EWA – Prosty system zarządzania bibliograficzną bazą danych dla mikrokomputera IBM PC/XT	1	13
SYSŁO MACIEJ M. – Maszyny i algorytmy równoległe		
Część 3. Maszyny i algorytmy systoliczne	1	6
TATAJ MARZENA, TORUŃ ALINA – Tomografia komputerowa	7	12

	nr	s.
TUCHOLSKI ADAM – NETBIOS – zasada działania i sposób użytkowania Część 2	1	20
TURSKI WŁADYSŁAW M. – Co należy i można zrobić	2	2
ZALEWSKI JANUSZ – Nie sama mikroelektronika...	2	5
ZALEWSKI JANUSZ – W poszukiwaniu programu rozwoju informatyki	2	16
ZALEWSKI JANUSZ – Rzetelność systemów komputerowych w zastosowaniach krytycznych	7	4
ZALEWSKI JANUSZ (oprac.) – Dynamiczne tworzenie zadań w Adzie	3	31
ZALEWSKI JANUSZ (oprac.) – Wielozasadowa wersja sita Erastotenesa w Adzie	4	26

Z KRAJU

Uchwała II Walnego Zjazdu Delegatów Polskiego Towarzystwa Informatycznego	2	7
Informacja '88 – **Jarosław Deminet, Hanna Włodarska**	2	19
W czterdziestą rocznicę powstania WNT	4	27
Stanowisko Polskiego Towarzystwa Informatycznego w sprawie ochrony prawnej programów komputerowych – opr. **Jacek Irlik**	7	21

ZE ŚWIATA

Perspektywy rozwoju systemu Unix – **J. R.**	1	25
Elastyczne środowisko programowania w Adzie – **M. Kuc**	1	29
Transoceaniczne kable światłowodowe – **J. R.**	3	IV okł.
Szybka i niezawodna drukarka diodowa – **J. R.**	3	IV okł.
Prognoza rozwoju pamięci na dyskach – **J. R.**	3	IV okł.
Biblioteki procedur Turbo Pascala – oprac. **M. Kuc**	4	29
Perspektywy rozwoju sieci przemysłowych – **J. R.**	4	IV okł.
Papier cyfrowy – nowy środek przekazywania informacji – **J. R.**	4	IV okł.
Światowy rynek mikroprocesorów 32-bitowych RISC – **J. R.**	4	IV okł.
Nowe kompilatory Ady dla mikroprocesorów – **MK, JZ**	5	29
Kolorowa drukarka laserowa – **J. R.**	5	IV okł.
Nowe modele PS/2 – **J. R.**	5	IV okł.
Oprogramowanie sieciowe 10-NET – **J. R.**	5	IV okł.
Francuskie mikrokomputery AT – **J. R.**	5	IV okł.
Nowa rodzina skomputeryzowanych stanowisk pracy – **J.R.**	5	IV okł.
Problemy stosowania Ady – oprac. **M. Kuc**	6	17
Turbo Pascal 4.0 – oprac. **M. Kuc**	6	20
Rocznica projektów standardów OSI – **J. R.**	6	22
Prognoza rozwoju drukarek – **J. R.**	6	22
Zastosowanie dysków optycznych – **J. R.**	6	22
Edukacja informatyczna w Japonii – **Maciej M. Sysło**	6	24
Wykonywanie programów DOS-a w systemie Unix – oprac. **D. Grabowicz**	7	23
Architektura dla Ady – oprac. **M. Kuc**	7	27
Pogromca konkurentów – dBase IV – oprac. **Wiktor Rzeczkowski**	7	28
Tajwan na rynku informatycznym **J. R.**	7	III okł.
Układy logiczne – **J. R.**	7	III okł.
Sieci MAP – **J. R.**	7	III okł.
Kostki RAM – **J. R.**	7	III okł.

Kto jest kim w IFIP

	nr	s.
James Finch – oprac. **M. K.**	4	30
Mike G. Rodd – oprac. **M. K.**	7	25

RECENZJE

Bazy danych – od koncepcji do realizacji – **Wojciech Olejniczak**	7	29

TERMINOLOGIA

Panie Redaktorze – **A. Kamieński**	1	III okł.
Językoznawcy o poprawności słownictwa informatycznego – **D. B.**	2	31
Słowniki, które warto wziąć do ręki	3	III okł.
Informatyka w szkołach okazją do uproszczenia ortografii – **Wojciech Mokrzycki**	4	III okł.
Słownik pojęć i terminów z dziedziny rozpoznawania i przetwarzania obrazów – **Wojciech Mokrzycki**		
Część 1	5	30
Część 2	6	32
Czym jest rzetelność systemu komputerowego? – **Janusz Zalewski**	7	30
Kod kreskowy – słowniczek podstawowych terminów – **J. Zal.**	8–12	III okł.

IMPREZY

Lokalne sieci komputerowe dla automatyzacji przemysłu	2	3
RELCOMEX '89	2	30
Komputery '89	3	12
Międzynarodowe Targi Wynalazków	3	12
Systems Science X	5	17
Wystawa Comdex	5	IV okł.
Grafika komputerowa, przetwarzanie obrazów	6	5
Informatyka w szkole	6	14
Komputeryzacja dydaktyki	6	14
18. konferencja GfI	6	28
Euromicro '89	6	31
Inżynieria wiedzy i systemy ekspertowe	7	3

LISTY

Szynowny Panie Redaktorze – **Maciej M. Sysło**	2	32
Od Redakcji – **Wacław Iszkowski**	2	32

NOWE KSIĄŻKI

Sztuczna inteligencja – **M. M.**	4	30
Arkusze obliczeniowe – **Janusz Zalewski**	6	4
Bazy danych – **W. Rzeczkowski**	6	30
Wydawnictwa Naukowo-Techniczne – **T. J.**	7	III okł.

3xTAG®

TAG Jeżeli **poszukujesz** programu, który umożliwi Ci pisanie tekstów w języku polskim, rosyjskim lub dowolnym innym języku europejskim

TAG Jeżeli **poszukujesz** programu, który umożliwi Ci pisanie dłuższych opracowań z użyciem wielu krojów czcionek na dowolnej drukarce

TAG Jeżeli **poszukujesz** programu, który zrobi za Ciebie korektę tekstu i wyszuka wyrazy bliskoznaczne

Wybierz edytor TAG

zdobywcę nagrody „Mikrolaur 89"
oryginalny produkt
Spółdzielni Pracy Informatyków
InfoService
GDAŃSK, Abrahama 1a
tel. 52-38-13 tlx 0512399

dla komputerów zgodnych z
IBM PC/XT/AT/386 • 512 KB RAM
• 1FDD • Hercules/CGA/VGA

InfoService®

EO/893/89

AKORD Sp. z o.o. oferuje:

16-bitowy przetwornik A/C dla IBM PC XT/AT

- **2 kanały 16-bitowe**
  zakres mierzonych napięć −5V +5V
  rozdzielczość 5 V/bit
  dynamika 10^6
  liniowość lepsza od 0,0015%
- **4 kanały 12-bitowe**
  zakres mierzonych napięć 0V +5V
  rozdzielczość 1mV/bit
  dynamika 10^4
  liniowość lepsza od 0,5%
- **8 wejść i 8 wyjść dwustanowych TTL**

System Komputerowego Wspomagania dla Chromatografii Gazowej dla IBM XT/AT V.IBM

- wyznaczanie początku, czasu retencji i końca piku
- integracja powierzchni piku z automatyczną korekcją linii zerowej
- możliwość automatycznego dobierania parametrów integracji
- rozdzielanie pików częściowo nałożonych
- możliwość automatycznego startu
- automatyczne zakończenie procesu chromatograficznego
- wyznaczenie współczynników równań regresji dla obliczeń ilościowych

Autonomiczny System Komputerowego Wspomagania dla Chromatografii Gazowej V.AUTO – funkcje identyczne jak w Systemie ... V.IBM, instalowany na sterowniku mikroprocesorowym.

Informacje i zamówienia: AKORD Sp. z o.o.
ul. Kasprowicza 15
81–852 Sopot
tel. 82-32-10
teleks 05-455-4 „Akces" PL

EO/931/89

TECHNIKA POMIAROWA, STEROWANIE, POMIARY MEDYCZNE I LABORATORYJNE

ambex®

ZAKŁAD ELEKTRONIKI

JEDNOSTKA INNOWACYJNO-WDROŻENIOWA

02-770 Warszawa, ul. Żabińskiego 7, tel. 24-15-69

oferuje

idealne do Twojego IBM PC XT/AT
oparte na elementach najwyższej światowej klasy

PRZETWORNIKI ANALOGOWO/CYFROWE i CYFROWO/ANALOGOWE 12-BITOWE

- precyzyjne wzmacniacze próbkująco-pamiętające
- pomiar analowy jednoczesny na wielu kanałach
- czasy przetwarzania przetworników a/c od 1 do 50 μs
- przetworniki a/c, c/a i we/wy sterujące na jednej płytce

SYSTEMY POMIAROWE „POD KLUCZ"

- tworzone przy udziale wybitnych specjalistów z różnych dziedzin
- IBM PC XT/AT z wysokiej klasy przetwornikami a/c i c/a
- wzmacniacze pomiarowe z izolacją galwaniczną sygnałów
- oprogramowanie zgodnie z wymaganiami klienta

WZMACNIACZE POMIAROWE

WZMACNIACZE Z IZOLACJĄ GALWANICZNĄ SYGNAŁÓW ANALOGOWYCH

OPROGRAMOWANIE SPECJALIZOWANE

UWAGA: naszym klientom zapewniamy bezpłatne oprogramowanie standardowe, wszechstronne konsultacje w okresie użytkowania sprzętu oraz serwis gwarancyjny i pogwarancyjny.

TECHNIKA POMIAROWA, STEROWANIE, POMIARY MEDYCZNE I LABORATORYJNE

EO / 276 / 89

Praca została wykonana w ramach Centralnego Programu Badawczo-Rozwojowego nr 8.13 *Budowa Krajowej Akademickiej Sieci Komputerowej*. Cel poznawczy „Formalne metody specyfikacji i konstruowania oprogramowania sieciowego".

**ZBIGNIEW HUZAR,
LUDWIK KUŹNIARZ
Centrum Obliczeniowe
Politechnika Wrocławska**

Język Lotos (1)

Specyfikacje typów danych

Język Lotos jest efektem prac nad formalnymi metodami specyfikacji oprogramowania sieciowego prowadzonych w ISO w ramach komitetu TC97 (w podkomitetach SC16 oraz SC21). Mimo iż został stworzony z myślą o konkretnych zastosowaniach (formalna specyfikacja modelu odniesienia ISO/OSI [1,5]), jest jednak językiem ogólnego przeznaczenia. Lotos jest narzędziem specyfikacji formalnej, tzn. umożliwia opisanie danego obiektu w sposób niesprzeczny, zwarty i precyzyjny, a także zupełny w tym sensie, że taki opis nie wymaga żadnej innej wiedzy o danym obiekcie. Język ten zawiera mechanizmy strukturalizacji tworzonych opisów (specyfikacji). Specyfikacje w Lotosie są specyfikacjami abstrakcyjnymi, tzn. język nie narzuca sposobu ich implementacji lub ograniczeń, a także pozwala pominąć nieistotne szczegóły.

Działanie wielu systemów informatycznych można opisać wyczerpująco stosując klasyczne podejście funkcyjne lub relacyjne, w których systemy są traktowane jako obiekty rozpoczynające działanie w pewnym stanie początkowym i kończące je osiągnąwszy pewien stan końcowy. Związek między stanem początkowym a stanem końcowym można wyrazić jako pewną funkcję lub – ogólniej – relację. Tymczasem występują systemy, w których nie chodzi o osiągnięcie pewnego stanu końcowego, lecz o to, aby bieżące działanie systemu rozumiane jako współdziałanie ze swym otoczeniem, było zgodne z określonym wzorcem. Przykładem takich systemów są systemy operacyjne, sieci komputerowe, systemy rezerwacji biletów itp. Przy specyfikacji tego rodzaju systemów, nazywanych *systemami reaktywnymi* [6], konieczne jest przyjęcie innego podejścia, nazwanego *podejściem behawioralnym*.

System reaktywny, widziany jako pojedynczy obiekt, prowadzi wymianę informacji ze swoim otoczeniem przez ustalone *kanały* komunikacyjne, zwane *bramkami*. Jednorazowe przesłanie informacji (zbioru wartości) przez bramkę nazywa się *interakcją*. Zewnętrzny obserwator (otoczenie) obserwując system reaktywny przez pewien okres zauważy ciąg interakcji. Specyfikacja systemu reaktywnego jest to definicja zbioru ciągów interakcji. Składa się ona z dwu zasadniczych części:
– specyfikacji abstrakcyjnych typów danych, służących do opisu elementów składowych obserwowanych ciągów interakcji,
– wyrażenia behawioralnego, opisującego wszystkie obserwowalne (możliwe do wystąpienia) ciągi interakcji.

Abstrakcyjne typy danych mają w literaturze wiele różnych modeli [3]. Do języka Lotos adaptowano podejście pochodzące z języka algebraicznych specyfikacji ACT ONE opracowanego w Uniwersytecie Technicznym w Berlinie Zachodnim pod kierunkiem H. Ehriga [2].

Opis zachowania systemów jest oparty na koncepcjach zaproponowanych przez R. Milnera z Uniwersytetu w Edynburgu, zawartych w CCS [6]. System jest traktowany jako pewien obiekt (zestaw równolegle działających procesów), który poprzez pewne punkty interakcji (bramki) oddziałuje na swoje otoczenie. Oddziaływanie polega na wymianie pewnego zestawu informacji między systemem a jego otoczeniem.

Zasadnicze elementy języka Lotos zostaną przedstawione w trzech kolejnych artykułach. Pierwszy zawiera wprowadzenie do specyfikacji abstrakcyjnych typów danych, drugi – omówienie wyrażeń behawioralnych, a trzeci – przykłady zastosowań. Pełny opis języka jest zawarty w [1] oraz [4]. Warto także dodać, że są już dostępne symulatory tego języka (np. symulator oferowany przez Uniwersytet Twente w Enschede).

TYP DANYCH JAKO ALGEBRA

Typ danych, czyli zbiór obiektów wraz z operacjami na tych obiektach, tworzy algebrę wielorodzajową. Klasycznie algebra rozumiana jest jako para $Alg = \langle N,F \rangle$, przy czym N jest pewnym ustalonym zbiorem elementów – *nośnikiem algebry*, a F zbiorem operacji algebry, tj. funkcji określonych na N. W algebrze wielorodzajowej N jest rodziną zbiorów, a argumenty i wartości funkcji należą do różnych zbiorów z N.
W języku Lotos definicja typu danych ma postać

```
type Nazwa-typu is                    (1)
  sorts Lista-nazw-rodzajów
  opns Lista-symboli-operacji
  eqns Lista-równości
endtype
```

i jest nazywana specyfikacją typu.
Listę tworzy ciąg elementów oddzielonych separatorami. *Nazwa rodzaju* jest to identyfikator pewnego zbioru wartości (dziedziny) występującego dalej w definicjach operacji. Symbol operacji ma postać

$f: a \rightarrow w$

gdzie f jest nazwą operacji, a jest napisem (być może pustym) złożonym z nazw rodzajów i określającym zbiór argumentów operacji, w jest nazwą rodzaju i określa zbiór wartości operacji. Dokładniej, symbol

$f: s_1 \ldots s_n \rightarrow s$

określa operację o nazwie f, która ma n argumentów – pierwszy ze zbioru o nazwie s_1, drugi ze zbioru o nazwie s_2 itd.; wynik operacji należy natomiast do zbioru o nazwie s.

SYGNATURA I TERMY

Sygnatura algebry jest to para $Syg = \langle S, OP \rangle$, gdzie S jest zbiorem nazw rodzajów (nośników) algebry, a OP – zbiorem symboli operacji. Określa ona strukturę algebry, tzn. liczbę nośników oraz „budowę" operacji. Wyznacza ona klasę algebr o danej sygnaturze, tzw. *Syg-algebr*. Algebra $A = \langle N,F \rangle$ jest *Syg*-algebrą, jeśli istnieją jednoznaczne odwzorowania h_1 ze zbioru S w N oraz h_2 ze zbioru OP w F, zachowujące strukturę, tzn. takie, że:

jeśli funkcja f jest postaci $f: s_1 \ldots s_n \rightarrow s$,
to $h_2(f) = g$, (2)
gdzie g jest funkcją z $h_1(s_1) \times \ldots \times h_1(s)$ w $h_1(s)$.

Dwie pierwsze składowe specyfikacji typu (algebry) określają właśnie sygnaturę pewnej rodziny typów (algebr). Na przykład specyfikacja

```
type NO is                            (3)
  sorts nat
  opns  z : -> nat
        nast : nat : -> nat
        plus : nat nat -> nat
endtype
```

definiuje rodzinę algebr, które składają się z jednego nośnika oraz trzech operacji – zeroargumentowej (stałej) o nazwie *z*, jednoargumentowej o nazwie *nast* oraz dwuargumentowej o nazwie plus. NO-algebrami są algebry A1, A2, A3:

$A1 = \langle N, \{0, p1, p2\}\rangle$

przy czym N jest zbiorem liczb naturalnych, 0 stałą zero, $p1$, $p2$ funkcjami takimi, że $p1\ (x) = x + 1$, $p2\ (x,y) = x + y$;

$A2 = \langle B, \{\underline{0}, |, \Rightarrow\}\rangle$

przy czym B jest zbiorem wartości logicznych $\underline{0}$, $\underline{1}$, | negacją logiczną, $\Rightarrow$ implikacją logiczną;

$A3 = \langle \{!\}^*, \{\varepsilon, c, \hat{}\}\rangle$

przy czym $\{!\}^*$ jest zbiorem skończonych ciągów złożonych z !, łącznie z ciągiem pustym ε, $\hat{}$ operacją konkatenacji dwu ciągów, a c operacją dołączenia elementu ! do zadanego ciągu, tj. $c\ (x) = x\hat{}!$.

Sygnatura określa właściwie abstrakcyjną składnię operacji wchodzących w skład pewnej klasy algebr. Pozwala także budować *termy*, które można uważać za abstrakcyjne wyrażenia. Termy definiuje się następująco. Niech $V(s)$ będzie zbiorem zmiennych rodzaju s (zbiorem obiektów oznaczających dowolną wartość rodzaju s). Zbiór $T(s)$ termów rodzaju s jest to najmniejszy zbiór o następujących właściwościach:

1) jeśli $x \in V(s)$, to $x \in T(s)$,
2) jeśli f jest nazwą operacji (stałej), taką że $f: \rightarrow s$, to $f \in T(s)$,
3) jeśli f jest nazwą operacji $f: s_1 \ldots s_n \rightarrow s$ oraz $t_i \in T(s_i)$ dla $i = 1,\ldots,n$, to $f(t_1,\ldots,t_n) \in T(s)$

Na przykład, dla typu NO jeśli przyjąć, że zmiennymi rodzaju *nat* są x oraz y, to termami rodzaju *nat* są między innymi *nast*(x), z, *plus*(x,y), *plus*(*nast*(z),*nast*(x)).

RÓWNOŚCI I SPECYFIKACJE

Termy są używane przy określaniu właściwości operacji w trzecim członie specyfikacji (1). Podaje się tam zestaw równości, które muszą być spełnione przez poszczególne operacje. Równością jest para termów połączona znakiem „=", przy czym określa się wcześniej rodzaj, do którego oba termy mają należeć, oraz zbiór zmiennych występujących w tych termach. Równość zapisuje się w Lotosie następująco:

forall *Lista-zmiennych*
  ofsort *Nazwa-rodzaju*
    $t_1 = t_2$

przy czym element *listy-zmiennych* jest postaci *nazwa:rodzaj*. Jeżeli w takiej definicji występuje kilka równości tego samego rodzaju lub występuje ten sam zbiór zmiennych, to te wspólne elementy umieszcza się raz przed właściwym ciągiem równości. Na wydruku 1 podano przykład specyfikacji zawierającej listę równości – jest to rozwinięcie przykładu (3).

```
type Nat is
  sorts nat
  opns  z : -> nat
        nast : nat : -> nat
        plus : nat nat -> nat
  eqns
   forall x:nat, y:nat
     ofsort nat
       plus(x,y) = plus(y,x)
       plus(nast(x),y) = nast(plus(x,y))
endtype
```

Wydruk 1

Specyfikacja z równościami określa precyzyjniej typ danych, tzn. ogranicza klasę algebr, które tej specyfikacji odpowiadają. Jeśli na przykład chodzi o zdefiniowanie typu danych odpowiadającego liczbom naturalnym z operacją dodawania, to specyfikację NO należy uznać za nie zadowalającą (zbyt mało precyzyjną), gdyż do klasy NO-algebr należy też algebra A2, którą trudno uznać za odpowiedni model dla liczb naturalnych.

Te intuicje związane ze zgodnością konkretnej algebry ze specyfikacją można formalnie wyrazić następująco. Specyfikacją jest trójka $Spec = \langle S, OP, R\rangle$, gdzie $Syg = \langle S, OP\rangle$ jest sygnaturą, a R – zbiorem równości. Niech $A = \langle N, F\rangle$ będzie Syg-algebrą. Oznacza to, że istnieją funkcje h_1 i h_2 o własności (2). Niech $V(s)$ będzie zbiorem zmiennych rodzaju s występujących w R, $V = \bigcup\{V(s) \mid s \in S\}$, $T(s)$ – zbiorem termów rodzaju s, a $T = \bigcup\{T(s) \mid s \in S\}$. Wartościowaniem zmiennych w algebrze A jest nazywane przekształcenie *val* ze zbioru V w N, takie że, jeśli $v \in V(s)$, to $val(v) \in h_1(s)$, tzn. zmiennym są przypisane wartości zgodnie z ich rodzajem. Wartościowanie zmiennych rozszerza się następnie na zbiór termów w następujący sposób:

1) jeśli t jest nazwą stałej f, gdzie $f: \rightarrow s \in \{op_i\}$, to $val(t) = h_2(f: \rightarrow s)$:
2) jeśli t jest postaci $f(t_1,\ldots,t_n)$, gdzie $f: s_1,\ldots,s_n \rightarrow s$,
   to $val(t) = h_2(f: s_1,\ldots,s_n \rightarrow s)\ (val(t_1),\ldots,val(t_n))$.

Równość r postaci $t_1 = t_2$ jest spełniona dla wartościowania *val* w algebrze A, jeśli $val(t_1)$ jest równe $val(t_2)$. Równość ta jest spełniona, jeśli jest spełniona dla każdego wartościowania. Zbiór równości jest spełniony w algebrze A, gdy jest spełniona każda równość z tego zbioru – i wtedy mówimy, że algebra A jest *Spec*-algebrą. Dla danej specyfikacji *Spec* może istnieć wiele *Spec*-algebr. Każdą z nich można przyjąć jako reprezentację tej specyfikacji, tzn. obiekt, który będzie używany do wykonywania obliczeń.

Dla omawianej uprzednio specyfikacji *Nat* jej reprezentacją może być zarówno algebra $A1$, jak i algebra $A3$. Specyfikacja *Nat* zawiera te właściwości obu algebr, które są dla nas istotne, dopuszczając ich różną konkretną realizację. Warto zauważyć, że termom *nast*(*nast*(z)) oraz *plus*(*nast*(z)) odpowiada w algebrze $A1$ element (liczba) 2, a w Algebrze $A3$ ciąg „!!"; dokładniej wartościami tych termów są 2 i !!. Na pewnym poziomie abstrakcji jest nieistotne to, jak dane operacje będą zrealizowane – czy jako operacje w arytmetyce całkowitej, czy operacje „na zapałkach" – ważne jest jedynie, aby między wynikami tych operacji zachodził określony związek.

STRUKTURALIZACJA SPECYFIKACJI

Podstawową techniką stosowaną przy tworzeniu złożonych obiektów, definicji lub opisów jest ich strukturalizacja, tzn. wydzielenie z danego obiektu części, które mogą tworzyć pewną całość, utworzenie ich, a następnie użycie ich przy tworzeniu obiektu złożonego. Język Lotos dostarcza dwu mechanizmów strukturalizacji specyfikacji typów danych.

Suma specyfikacji

Na wydruku 2 podano przykład definicji typu. Jest to specyfikacja typu danych, w których występują dwa nośniki, a także „mieszana" operacja do przekształcania wartości różnych rodzajów. W tej specyfikacji można wyodrębnić prostsze typy tworzące pewną całość oraz część, w tym wypadku jedną operację, która je w określony sposób wiąże. Powyższą definicję można w Lotosie zapisać w sposób modularny, odzwierciedlający jej strukturę; zapewnia to operacja składania specyfikacji typów – konstrukcja sumy typów. Sumę typów zapisuje się w taki sposób, że po słowie **is** w (1) podaje się listę nazw typów. Oznacza to, że specyfikację typów o podanych tam nazwach należy włączyć do danej definicji w taki sposób, że tworzy się sumę mnogościową wszystkich definicji.

```
type NatLog is
  sorts nat, log
  opns  z : -> nat
        nast : nat : -> nat
        plus : nat nat -> nat
        p : -> log
        neg : log -> log
        impl : log log -> log
        mr  : nat nat -> log
  eqns
   forall x:nat, y:nat, a:log, b:log
    ofsort nat
       plus(x,y) = plus(y,x)
       plus(nast(x),y) = nast(plus(x,y))
    ofsort log
       neg( neg( b ) ) = b
       impl( p,p ) = p
       impl( p,neg(p) ) = neg(p)
       impl( neg(p),a ) = p
       mr( z,x ) = p
       mr( nast(x),nast(y)) = mr(x,y)
endtype
```

Wydruk 2

W ostatnim przykładzie można więc najpierw podać specyfikację typu *Log*, a następnie, korzystając także z (3), zdefiniować typ *NatLog* (wydruk 3).

```
type Log is
  sorts  log
  opns     p : -> log
         neg : log -> log
        impl : log log -> log
  eqns
   forall a:log, b:log
    ofsort log
       neg( neg( b ) ) = b
       impl( p,p ) = p
       impl( p,neg(p) ) = neg(p)
       impl( neg(p),a ) = p
endtype

type NatLog is Nat, Log
  opns    mr  : nat nat -> log
  eqns
   forall x:nat, y:nat, a:log, b:log
    ofsort log
       mr( z,x ) = p
       mr( nast(x),nast(y) ) = mr(x,y)
endtype
```

Wydruk 3

Specyfikacje z wydruków 2 i 3 są sobie równoważne, tzn. definiują ten sam typ. Należy jednak zwrócić uwagę na to, iż definiując *NatLog* w ten sposób określa się jeszcze dwa typy, które mogą być wykorzystywane dalej niezależnie od *NatLog*. Ponadto, co jest być może ważniejsze, na typ *NatLog* została nałożona pewna struktura.

Warto także zwrócić uwagę na to, iż typy *Nat* i *Log* mają sygnatury identyczne z dokładnością do użytych nazw (tj. izomorficzne), lecz różnią się zasadniczo tym, że ich operacje mają inne właściwości.

Specyfikacje sparametryzowane

W przypadku bardziej złożonych typów, np. tablic, plików, stosów lub kolejek, okazuje się często, że ogólna struktura tworzących je obiektów nie zależy od elementów składowych. W Lotosie do wyrażania tego faktu służy specyfikacja w postaci typu sparametryzowanego. Ma ona postać

```
type Nazwa-typu is

  formalsorts Lista-nazw-rodzajów
  formalopns  Lista-symboli-operacji
  formaleqns  Lista-równości

  sorts Lista-nazw-rodzajów
  opns  Lista-symboli-operacji
  eqns  Lista-równości

endtype
```

Pierwsze trzy elementy tej definicji tworzą tzw. część formalną, zawierającą charakterystykę tych elementów, od których w danym momencie chce się abstrahować. Specyfikację typu uzyskuje się ze specyfikacji sparametryzowanej podstawiając konkretne, wcześniej zdefiniowane obiekty (rodzaje i operacje) za odpowiednie obiekty formalne. Konstrukcja aktualizacji ma postać:

type *Nazwa-typu* **is** *Nazwa-typu-sparametryzowanego*
actualizedby *Nazwa-typu* **using**
Lista-podstawień-rodzajów
Lista-podstawień-operacji
endtype

przy czym *podstawienie-rodzaju* ma postać

sortnames *nazwa-rodzaju* **for** *nazwa-rodzaju-formalnego*

a *podstawienie-operacji* ma postać

opnnames *nazwa-operacji* **for** *nazwa-operacji-formalnej*

Dobrą ilustracją jest tutaj specyfikacja stosu. Najpierw specyfikuje się stos niezależnie od typu jego elementów (wydruk 4).

```
type Stos  is
  formalsorts elem
  formalopns  e0 -> elem
  sorts  stos
  opns
        pusty : -> stos
        odczyt : stos -> elem
        ze_stosu : stos -> stos
        na_stos  : stos elem -> stos
  eqns
    forall e : elem, s : stos
      ofsort elem
        odczyt( pusty ) = e0
        odczyt( na_stos(s,e) )  = e
      ofsort stos
        ze_stosu( pusty ) = pusty
        ze_stosu( na_stos(s,e) ) = s
endtype
```

Wydruk 4

Stos liczb naturalnych można uzyskać podając specyfikację

```
type StosNat is Stos
    actualizedby Nat using
    sortnames nat for elem
    opnnames z for e0
endtype
```

Warto zauważyć, że stała *z* występująca w *Nat* pełni w typie *StosNat* funkcję sygnalizatora błędu, tzn. odczyt ze stosu pustego daje w wyniku wartość *z*.

Część formalną specyfikacji sparametryzowanej można uważać za zestaw właściwości (ograniczeń), jakie ma posiadać typ aktualizujący taką specyfikację. Typ aktualizujący musi mieć zatem odpowiednie nośniki oraz operacje i to takie, których właściwości implikują właściwości wyrażone przez właściwości formalne. Tę część specyfikacji sparametryzowanej można więc uważać za swego rodzaju filtr. Na przykład, dla specyfikacji *GrupaF* z wydruku 5 dopuszczalna jest aktualizacja *Nat_GrupaF*, lecz aktualizacja *Log_GrupaF* jest niepoprawna.

```
type GrupaF  is
  formalsorts x
  formalopns  i : -> x
              f : x x -> x
  formaleqns
     forall  a,b,c : x
       ofsort  x
         f(a,i) = a
         f( a,f(b,c)) = f( f(a,b),c )
endtype


type Nat_GrupaF is GrupaF
    actualizedby Nat using
    sortnames nat for x
    opnnames   z for i
           plus for f
endtype


type Log_GrupaF is GrupaF
    actualizedby Log using
    sortnames log for x
    opnnames   t for i
           impl for f
endtype
```

Wydruk 5

Pozostała część specyfikacji sparametryzowanej określa te elementy, które mają być dołączone do typu aktualizującego, w celu utworzenia nowego typu. Jeśli zatem zostanie zdefiniowany typ *Grupa*, to aktualizacja *Nat_Grupa* spowoduje utworzenie typu identycznego z typem *Nat_Grupa* (wyduk 6).

```
type Grupa  is
  formalsorts x
  formalopns  i : -> x
              f : x x -> x
  eqns
     forall  a,b,c : x
       ofsort  x
         f(a,i) = a
         f( a,f(b,c)) = f( f(a,b),c )
endtype


type Nat_Grupa is Grupa
    actualizedby NO using
    sortnames nat for x
    opnnames   z for i
           plus for f
endtype


type Nat_Grupa is
   sorts nat
   opns  z : -> nat
         nast : nat : -> nat
         plus : nat nat -> nat
endtype
 eqns
    forall  a,b,c : x
      ofsort  x
        plus(a,i) = a
        plus( a,plus(b,c)) = plus( plus(a,b),c )
endtype
```

Wydruk 6

PRZEMIANOWYWANIE

Język Lotos jest wyposażony także w konstrukcję zmiany nazw. Jest ona użyteczna wówczas, gdy w nowym środowisku chce się wykorzystać wcześniej zdefiniowane typy bez zmiany ich znaczenia, lecz nadając im nową „szatę graficzną", lepiej wyrażającą intuicje związane z danym

typem i jego operacjami. Ręczne przeprowadzenie takich zmian, zwłaszcza dla dłuższych definicji, jest uciążliwe i często prowadzi do błędów. Przykładowo do zdefiniowania kolejki LIFO dla potrzeb systemu zarządzania procesami można wykorzystać specyfikację typu *Stos*, przedefiniowując ją w następujący sposób

```
type LIFO is Stos
  renamedby
  formalsortnames zadanie for elem
  formalopnames brak zadan for er
  sortsnames lifo for stos
  opnames czekaj    for na_stos
  opnames startuj   for ze_stosu
  opnames sprawdz   for odczyt
endtype
```

NOTACJA INFIKSOWA I SPECYFIKACJE WARUNKOWE

Dla operacji dwuargumentowych można używać notacji infiksowej. Aby zdefiniować operację jako infiksową, należy definiując symbol operacji jej nazwę ująć w znaki podkreślenia, tzn. znaki te umieścić na początku i końcu nazwy. Na przykład dla symbolu operacji

_mr_ : nat nat → log
można używać zapisu *x mr y*.

Ponadto nazwę operacji może tworzyć dowolny ciąg znaków (z wyjątkiem pewnych znaków zastrzeżonych, jak przecinek, średnik, dwukropek). Można zatem zdefiniować na przykład operację

_+_ : nat nat nat
i używać „naturalnego" zapisu $x + y$.

Przy definiowaniu właściwości operacji można także używać *równości warunkowych*. Opis tego mechanizmu można znaleźć w [1, 4]; w niniejszym opracowaniu, mającym na celu zaprezentowanie podstawowych idei związanych z definiowaniem abstrakcyjnych typów danych języka Lotos, został on pominięty ze względu na konieczność wprowadzenia przy jego omawianiu dodatkowych pojęć z dziedziny systemów formalnych.

SPECYFIKACJE STANDARDOWE

W języku Lotos jest dostępna pewna liczba pierwotnie definiowanych typów standardowych. Są one umieszczone w tzw. bibliotece. Aby skorzystać z tych typów, należy podać następującą specyfikację

library *lista identyfikatorów typów standardowych*
endlib

```
type Boolean is
sorts Bool
opns true, false : -> Bool
     not           -> Bool -> Bool
     and , or , xor , implies ,
     iff , eq , ne                : Bool Bool -> Bool
endtype

type BasicNaturalNumber is
sorts Nat
opns 0    : -> Nat
     succ : Nat -> Nat
     + , * , ** : Nat Nat -> Nat
endtype

type NaturalNumber is BasicNaturalNumber, Boolean
opns  eq , ne , lt , le , ge , gt  : Nat Nat -> Nat
endtype

type DecNatRepr is DecString
opns NatNum : DecString -> Nat
endtype

type DecString is NonEmptyString actualizedbyDecDigit using
sortnames  DecDigit for Element
           Bool    for FBool
           DecString for NonEmptyString
opnames    Dec  for String
endtype

type DecDigit is Boolean, NaturalNumber
sorts DecDigit
opns  0,1,2,3,4,5,6,7,8,9 : -> DecDigit
   eq , ne , lt , le , ge , gt  : DecDigit DecDigit -> Bool
   NatNum : DecDigit -> Nat
endtype
```

Wydruk 7

dokończenie na III stronie okładki

Opam KATOWICE

PRZEDSIĘBIORSTWO PROJEKTOWANIA DOSTAW
I REALIZACJI OBIEKTÓW OCHRONY POWIETRZA

Opam

40-952 Katowice, ul. Lubuska 23

sprzeda

MINIKOMPUTER MERA–60 z procesorem M–2
w następującej konfiguracji:

- stacja dysków elastycznych SP–60 M
- stacja dysków twardych 5 MB typ SM 5400 – 2 szt.
- pamięć taśmowa typ SM 5300.01
- pamięć buforowa typ PPZ–01
- monitory – 4 szt.
- drukarkę D–100

Informacji udziela dział TI, tel. 58-80-61 (do 4) wewn. 244, 247.

EO/944/89

ul. Astrów 7,
40-045 Katowice,

tel. 518 628,
teleks 315687 comp pl

faks: 518 628

ZAKŁAD SYSTEMÓW MIKROKOMPUTEROWYCH

Systemy teletransmisji ODRA/ICL 1900, 2900, 39, ME 29:

- skaner MPXSCAN-8087
- procesor sieci teletransmisji danych MICOMP 8075 (emulacja ICL 750)
- program teletransmisji danych MICROSS-FXBM (IBM PC, rozproszone bazy danych, wersja sieciowa)
- adaptery, testery

Kilkadziesiąt instalacji w największych systemach na terenie całego kraju!

Integracja systemów ICL/ODRA – PC, NOVELL, XENIX

Systemy wielodostępne: supermikrokomputery ALR + UNIX!

EO/1124/89

Okres istnienia oprogramowania

dokończenie ze s. 14

Wydaje się, że ich zawartość może w chwili obecnej stanowić punkt wyjścia do opracowania analogicznych, lecz chyba ważniejszych zaleceń dla przygotowania warunków umowy (ang. *terms of reference*) dotyczącej studium, opartych na trzech podstawowych pytaniach:

- Co należy zrobić wykonując studium realizowalności?
- Jak to zrobić?
- Jak zweryfikować otrzymane wyniki?

LITERATURA

[1] Gannon J.D., Katz E.E., Basili V.R.:Metrics for Ada Packages – An Initial Study. Communications of the ACM, Vol. 29, No. 7, pp. 616–623, 1986

[2] Gasik S.: Wprowadzenie do programometrii. Informatyka, s. 9–13, nr 11–12, 1985

[3] IAEA: Manual on Quality Assurance for Computer Software. Final Version, International Atomic Energy Agency, Vienna, April 1987

[4] IEEE Standard Glossary of Software Engineering Terminology. IEEE Std 729–1989

[5] IEEE Standard for Software Quality Assurance Plans. IEEE Std 730–1989

[6] IEEE Standard Dictionary of Measures to Produce Reliable Software. IEEE Std. 982.1–1988

[7] IEEE Guide to Standard Dictionary of Measures to Produce Reliable Software. IEEE Std. 982.2–1988

[8] IEEE Guide to Software Quality Assurance Planning. IEEE Std 983–1986

[9] IEEE Standard for Software Life Cycle Processes. IEEE Project 1074

[10] MUSE Project: A list of references on metrics. Internal Report. ESPRIT Project No. 1257, Brussels, 1989

[11] Redmill F. (Ed.): Dependability of Critical Computer Systems. Vol. 2. Elsevier Applied Science, London, 1989

[12] Redmill F., Barber D.R.: A Guideline for Conducting a Feasibility Study and Preparing the Feasibility Study Report. Document SD0030, British Telecomm International, dept. NPD/CA, London, December 1986.

„Technika Zagraniczna"
magazyn techniczno-gospodarczy

Wydawnictwo Czasopism i Książek Technicznych NOT SIGMA wznawia po dziewięcioletniej przerwie wydawanie czasopisma „Technika Zagraniczna" – tym razem w postaci dwumiesięcznika (format pisma A4) o objętości i nakładzie zależnym od zapotrzebowania.

Pierwszy numer „TZ" (magazyn techniczno-gospodarczy) ukaże się pod koniec pierwszego kwartału 1990 r.

Proponujemy na naszych łamach szeroki zakres tematyczny, który umożliwi śledzenie przemian zachodzących w technice i gospodarce światowej, w obrębie działów:

- budowa maszyn i urządzeń
- elektronika, elektrotechnika, komputeryzacja
- budownictwo
- ochrona środowiska
- technika w rolnictwie i przemyśle spożywczym
- chemia
- informacje techniczno-gospodarcze.

Czasopismo „Technika Zagraniczna" jest adresowane do wielu odbiorców, a przede wszystkim do: inżynierów wszystkich branż przemysłowych, kadry kierowniczej dużych i małych firm państwowych i prywatnych, decydentów różnych szczebli, menadżerów i pracowników handlu zagranicznego, specjalistów z zakresu marketingu.

Redakcja „TZ" zachęca do prenumeraty czasopisma zakłady pracy i osoby prywatne zainteresowane uzyskaniem dostępu do informacji na temat najnowszych osiągnięć w technice światowej.

Wstępna cena pojedynczego egzemplarza czasopisma na rok 1990 wynosi 5000 zł. Tak więc zaliczkowa wpłata na prenumeratę półroczną wyniesie 15 000 zł, a roczną 30 000 zł. W przypadku konieczności zmiany ceny spowodowanej rosnącymi stale kosztami, wymagana będzie dopłata.

Wpłaty na prenumeratę na II półrocze będą przyjmowane do 31 maja 1990 r. na konto SIGMY.

Dodatkowych informacji na temat czasopisma udziela redakcja: Technika Zagraniczna, ul. Mazowiecka 12, 00-950 Warszawa, tel. 27-43-65.

MAREK GONDZIO
MAREK PAWŁOWSKI
JACEK WYTRĘBOWICZ
Instytut Informatyki
Politechnika Warszawska

Uruchamianie urządzeń mikroprogramowanych (2)

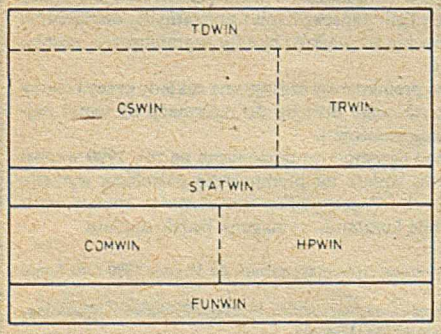
W poprzedniej części przedstawiono podstawowe problemy związane z uruchamianiem urządzeń mikroprogramowanych, a także scharakteryzowano – od strony sprzętowej – system MIDES wspomagający uruchamianie tych urządzeń, zrealizowany w Instytucie Informatyki Politechniki Warszawskiej. W tej części artykułu omówiono część programową systemu Mides.

KOMUNIKACJA SYSTEMU Z OPERATOREM

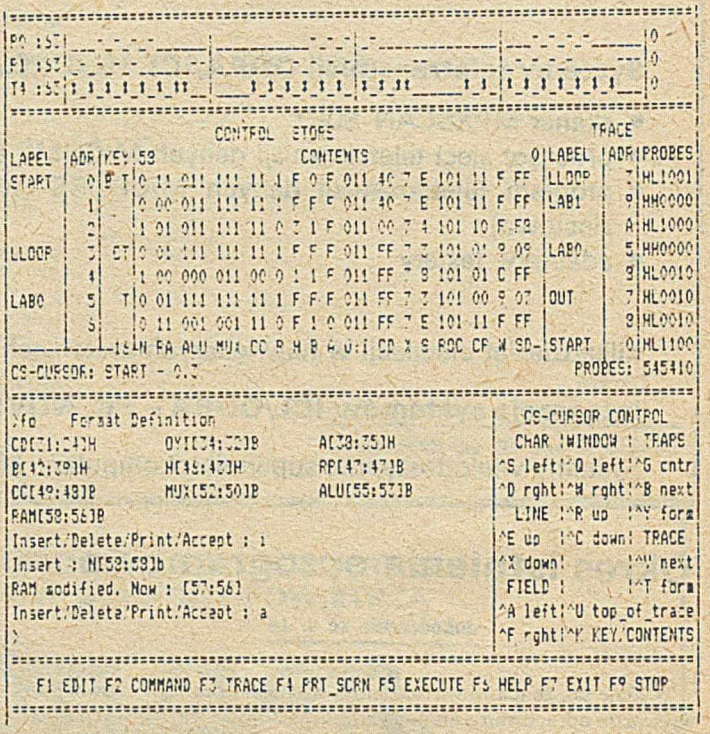
Do efektywnego sterowania procesem uruchamiania mikroprogramu operatorowi są niezbędne liczne informacje o bieżącym stanie prototypu, kodach wykonywanych mikrorozkazów, bieżącej zawartości pamięci śladu itp. Jedynym miejscem, w którym informacje te mogą być przedstawione operatorowi, jest ekran monitora IBM PC. W celu czytelnego wyświetlania dużej liczby danych (w dodatku różnych rodzajów), ekran podzielono na okna o ustalonych przeznaczeniach. Podział ekranu na okna nie jest jednak stały, zależy bowiem od stanu MIDES-a i wymagań operatora, który w ograniczonym zakresie może sterować organizacją ekranu.

Do obsługi ekranu służy specjalny moduł programu *SUM*, który sam dostosowuje się do zainstalowanej w danym mikrokomputerze karty graficznej. Dzięki temu na monitorach kolorowych poszczególne okna mogą różnić się kolorami, co zwiększa czytelność wyświetlanych danych. Moduł ten zawiera specjalny mechanizm chroniący przed męczącym dla wzroku migotaniem, występującym podczas przemieszczania informacji na ekranie IBM PC wyposażonego w kartę graficzną CGA. Dodatkowo moduł ten wyposażono w funkcję drukowania na drukarce bieżącej zawartości ekranu (tzw. trwałej kopii). Funkcja ta formatuje wyświetlane dane, gwarantując odpowiednią czytelność kopii ekranu na papierze. W odróżnieniu od podobnej funkcji PC-DOC (uaktywnianej klawiszem *PrtSc*), drukowanie trwałej kopii przez *SUM* odbywa się w tle, bez blokowania użytkownikowi dostępu do MIDES-a. Przykład trwałej kopii ekranu MIDES-a przedstawiono w wydruku.

Ogólnie ekran MIDES-a jest podzielony na cztery okna nazywane stałymi, ponieważ są cały czas wyświetlane. W dwu z tych okien mogą być otwierane tzw. okna tymczasowe. Maksymalnie ekran MIDES-a może być podzielony na siedem okien (rys.), przy czym niektóre okna mogą być dodatkowo podzielone na podokienka. Poszczególne okna mają poniżej określone przeznaczenia.



Podział ekranu MIDES-a na siedem okien



Trwała kopia ekranu MIDES-a

● **Okno pamięci sterującej** (*CSWIN*) zawiera opis kilku (do 10) mikrorozkazów. Każdy wiersz okna jest przeznaczony dla jednego mikrorozkazu.

Użytkownik sam definiuje format słowa mikrorozkazowego (podział słowa na pola) oraz decyduje, które pola i w jakiej kolejności mają być wyświetlane w tym oknie. Definiując format, użytkownik nadaje poszczególnym polom nazwy, których może potem używać przy odwołaniach do pamięci sterującej. Nazwy pól znajdujących się w oknie *CSWIN* są wyświetlane w najniższym wierszu tego okna. Opis każdego mikrorozkazu zawiera także:

– adres (podokienko *ADR*),
– etykietę (podokienko *LABEL*), jeżeli została zdefiniowana w mikroprogramie, a tablica symboli mikroprogramu została udostępniona MIDES-owi,
– wartości znaczników przypisanych danemu mikrorozkazowi (podokienko *KEY*).

Znacznik *T* wskazuje mikrorozkazy, których wykonanie ma być zapamiętane w pamięci śladu, znacznik *B* – mikrorozkazy, w których ustawiono pułapki adresowe, a *C* – mikrorozkaz, w którym jest ustawiona pułapka z licznikiem (wartość licznika jest wyświetlana w najniższym wierszu kolumny *KEY*). Operator ma możliwość przeglądania w oknie *CSWIN* treści źródłowej mikroprogramu, zarówno w kolejności umieszczania go w pamięci, jak również w kolejności wykonywania mikrorozkazów. Funkcja ta znacznie zwiększa efektywność procesu uruchamiania.

● **Okno konwersacyjne** (*COMWIN*) służy do dialogu użytkownika z MIDES-em. W oknie tym są wyświetlane komunikaty systemu oraz wprowadzane przez użytkownika polecenia dla MIDES-a.

● **Okno pamięci śladu** (*TRWIN*) służy do przedstawienia zawartości wskazanego fragmentu pamięci śladu w odpowiednio zdekodowanej postaci. Stany sond dwuwartościowych są reprezentowane przez 0 i 1, a stany sond trójwartościowych przez *L*, *H* i *Z*. Najniższy wiersz okna zawiera stan sond adresowych i śladu, przechwycony dokładnie w momencie zawieszania wykonywania mikroprogramu. W podokienku *PROBES* są wyświetlane tylko stany aktualnie aktywnych sond śladu.

● **Okno diagramów czasowych** (*TDWIN*) przedstawia zmiany stanów aktywnych sond śladu (zarejestrowane w pamięci śladu) w postaci diagramów czasowych. MIDES umożliwia wyświetlanie do czterech diagramów czasowych. Na rysunku przedstawiono zmiany stanu sond dwuwartościowych *P0* i *P5* oraz sondy trzywartościowej *T1* (* oznacza stan niezdefiniowany.)

● **Okno informacji pomocniczych** (*HPWIN*) służy do wyświetlania skróconych informacji o głównych funkcjach MIDES-a, parametrach poleceń, trybach pracy itp. Wiele z tych opisów ma postać menu. W kopii ekranu przedstawionej na wydruku okno HPWIN zawiera opis klawiszy sterujących ruchem kursora w oknie CSWIN.

● **Okno stanu bieżącego** (*STATWIN*) stanowi wiersz ekranu, w którym MIDES wyświetla pewne elementy składające się na stan bieżący systemu (np. numery sond aktywnych i położenie kursora w pamięci sterującej).

● **Okno opisu klawiszy funkcyjnych** (*FUNWIN*) to najniższy wiersz ekranu, w którym są podane znaczenia poszczególnych klawiszy funkcyjnych klawiatury IBM PC w systemie MIDES.

Okna *CSWIN*, *STATWIN*, *COMWIN* i *FUNWIN* są zawsze wyświetlane, okno *HPWIN* – o ile występuje – przysłania część okna *COMWIN*, analogicznie *TRWIN* pokrywa część okna *CSWIN*. Okno diagramów czasowych zajmuje zawsze całą szerokość ekranu i pokrywa jednakową liczbę wierszy w oknach *CSWIN* i ewentualnie *TRWIN* (rys.).

Kolejnym ważnym czynnikiem wpływającym na sposób komunikowania się operatora z MIDES-em jest duża liczba rozmaitych poleceń systemowych o różnorodnych parametrach. System ma przeznaczenie profesjonalne, powinien więc umożliwiać szybkie wprowadzanie poleceń systemowych. Dla doświadczonego użytkownika jest przeznaczony tradycyjny sposób symbolicznego wprowadzania poleceń i ich parametrów w jednym wierszu dyrektywy systemowej. Natomiast z myślą o początkujących użytkownikach MIDES został wyposażony w alternatywny sposób wprowadzania poleceń, tzw. metodę zapytań z domniemanymi odpowiedziami. Wprowadzając polecenie użytkownik sam decyduje, która z tych metod jest dla niego wygodniejsza.

Niektóre bezparametrowe funkcje MIDES-a mogą być wywołane także przez naciśnięcie odpowiednich klawiszy funkcyjnych IBM PC. Znaczenie tych klawiszy wyjaśnia okno *FUNWIN*.

PRZEGLĄD POLECEŃ SYSTEMOWYCH

Funkcjonalne możliwości MIDES-a są określone przez parametry sprzętowe systemu *SUMUS*, z których główne przedstawiono w poprzedniej części artykułu, ale przede wszystkim możliwości MIDES-a są reprezentowane przez zestaw poleceń udostępnianych użytkownikowi. Liczba poleceń systemowych i ich wariantów jest duża i pełne jej przedstawienie przekracza ramy tego artykułu, zostaną więc krótko scharakteryzowane jedynie podstawowe grupy poleceń systemowych.

Definiowanie trybów pracy MIDES-a

Polecenia należące do tej grupy umożliwiają:
– określenie rozmiaru pamieci sterującej dla danego prototypu (tzn. liczby słów i liczby bitów w słowie),
– specyfikację zegara, która obejmuje określenie źródła, umiejscowienie generatora, okres, czas trwania fazy wysokiej i niskiej, częstotliwość i wypełnienie (operator wybiera najdogodniejszą dla niego metodę definiowania zegara),
– dołączenie i odłączenie rejestru na wyjściu pamięci,
– określenie formatu (podziału słowa mikrorozkazowego na pola): przykład tego polecenia przedstawiono na wydruku.

Edycja pamięci sterującej

Ta grupa poleceń umożliwia ładowanie zawartości pamięci sterującej, wyprowadzanie tej zawartości na nośniki zewnętrzne, a także przeglądanie, przeszukiwanie i ewentualną modyfikację tej zawartości. Informacje dotyczące zawartości pamięci sterującej są wyświetlane w oknie *CSWIN*, przy czym operator sam określa, które mikrorozkazy mają być wyświetlane. Pojedynczych zmian zawartości pamięci można dokonać tylko przez modyfikację zawartości okna *CSWIN*. Sterowanie kursorem w *CSWIN*, a tym samym możliwości przeglądania zawartości pamięci, są oparte na zasadach znanych z edytora Wordstar (listę dostępnych operacji zawiera okno *HPWIN* na wydruku.

Polecenia należące do tej grupy umożliwiają m.in.:
– przesyłanie danych między nośnikiem zewnętrznym a pamięcią sterującą (w obu kierunkach),
– określanie pól do wyświetlania w oknie *CSWIN* oraz ich wzajemnego położenia na ekranie,
– poszukiwanie zdefiniowanego wzorca we wskazanym obszarze pamięci sterującej.
– wypełnianie fragmentu pamięci określonym wzorcem,
– wprowadzenie nowego słowa pamięci do mikroprogramu,
– usunięcie wskazanego mikrorozkazu.

Przeglądanie pamięci śladu

W normalnym trybie pracy MIDES-a do pamięci śladu są wpisywane stany sond śladu oraz adresy tych mikrorozkazów, dla których jest ustawiony wskaźnik śledzenia (znacznik *T* w polu *KEY* w oknie *CSWIN* – patrz wydruk). Operator może jednak zażądać pełnego zbierania śladu, tzn. z wykonania wszystkich mikrorozkazów, niezależnie od stanu ich wskaźników śledzenia.

Do prezentacji zawartości śladu służą okna *TDWIN* i *TRWIN*. W obu oknach są wyświetlane tylko informacje o aktualnie aktywnych sondach (dołączonych do konkretnych punktów prototypu).

Funkcje poleceń tej grupy obejmują:
– wyrób trybu śledzenia (pełny lub selektywny – ze wskazaniem mikrorozkazów do śledzenia),
– określenie maski dla selektywnego przeglądania pamięci śladu; zdefiniowanie maski umożliwia operatorowi wybranie do wyświetlania tylko części słów zapisanych w pamięci śladu (np. mikrorozkazów o adresach należących do pewnego zakresu lub słów, w których pewne sondy mają ustaloną, określoną wartość),
– specyfikację sond, których stany mają być dodatkowo wyświetlane w postaci diagramów czasowych,
– wydrukowanie na drukarce zawartości całej pamięci śladu w ustalonym formacie.

Obsługa pułapek

W systemie MIDES terminem *pułapki* określa się mechanizmy zatrzymywania działania mikroprogramu, którymi może sterować operator. Do mechanizmów tych należą:
– pułapki adresowe, które w postaci wskaźnika pułapki (znacznik *B* w polu *KEY*) mogą być przypisane dowolnym mikrorozkazom,
– pułapka z licznikiem (mikrorozkaz jest wskazywany przez znacznik *C* w polu *KEY*, a wartość licznika jest wyświetlana w najniższym wierszu pola *KEY*),
– pułapki warunkowe definiowane w postaci wyrażeń opisujących stany sond śladu, dla których ma nastąpić zatrzymanie wykonywania mikroprogramu.

Obsługa pułapek obejmuje polecenia ustawiania i usuwania pułapek oraz wyświetlania listy pułapek ustawionych. Adresy pułapek zwykłych i adres pułapki z licznikiem można wskazywać również przez odpowiednią modyfikację zawartości podokienka *KEY*.

Sterowanie wykonaniem mikroprogramu

Wykonanie mikroprogramu może nastąpić tylko w wyniku polecenia operatorskiego. Operator może polecić:
– rozpoczęcie wykonania mikroprogramu od wskazanego mikrorozkazu,

– kontynuowanie mikroprogramu od miejsca zatrzymania,
– krokowe wykonanie wskazanej liczby mikrorozkazów.
Zatrzymanie mikroprogramu następuje po wykryciu przez analizator zdarzeń jednego ze zdarzeń, wymienionych w opisie bloku analizatora zdarzeń (por. w poprzedniej części artykułu). Operator wpływa na warunki zatrzymania mikroprogramu przez wprowadzenie pułapek lub przez wydanie polecenia asynchronicznego zatrzymania mikroprogramu.

Polecenia dodatkowe

Do grupy tej należą polecenia:
– testowania sprawności urządzenia,
– wydruku trwałej kopii ekranu na drukarce,
– przeglądania skróconej dokumentacji systemu (listy poleceń),
– powrotu do systemu operacyjnego.
Projekt urządzenia jest otwarty i umożliwia dołączenie do systemu generatora wymuszeń i emulatorów typowych sekwenserów i procesorów segmentowych.

* * *

Uruchamianie urządzeń mikroprogramowanych jest bardzo trudne bez narzędzi odpowiednich do tego celu. Trudne jest również ich wytworzenie, ze względu na duży koszt przy ograniczonej liczbie ich odbiorców. Realizacje uproszczonych systemów wspomagających uruchomienie układów mikroprogramowanych prowadzą do pojawienia się namiastek tego rodzaju systemu. Przykład takiego podejścia przedstawiono na konferencji MICRO 16 w 1983 r. [2]. System Logic Engine Development pozwala na uruchomienie układów sterowania z sekwenserem Am2910 i ma bardzo skromne możliwości. Trudno go uważać za system profesjonalny.

W Instytucie Informatyki PW od kilkunastu lat jest prowadzone nauczanie mikroprogramowania. Wykład poświęcony tej tematyce jest wspomagany przez laboratorium. Przez kilka lat był w nim używany system *SUMUS* [1], który pomógł zebrać doświadczenia niezbędne do odbudowy systemu *MIDES*. System *MIDES* został zaprojektowany z rozmachem stawiającym go w rzędzie najlepszych systemów wspomagających uruchamianie urządzeń mikroprogramowanych. Czego przykładem jest najnowszy produkt firmy Step Engineering, system *STEP 50*, młodszy od prezentowanego tu systemu *MIDES*. Obydwa systemy mają identyczną architekturę i porównywalne właściwości funkcjonalne.

LITERATURA

[1] Budkowski S., Woźniak A.: System uruchomieniowy mikroprogramowanych układów sterowania, Materiały IV Krajowej Konferencji Naukowo-Technicznej SEP, Warszawa, 13 września 1984
[2] Prosser F., Winkel D.: The Logic Engine Development. Proc. MICRO-16, ACM SIGMICRO Newsletter, Vol. 14, No. 4, December 1983
[3] The Am2900 Family Data Book with Related Support Circuits. Advanced Micro Devices, 1978.

PZ Anglomet

oferuje usługi w zakresie:
regeneracji taśm barwiących do drukarek

Informacje i zamówienia
Biuro Inter-Vip, Warszawa, tel. 10-30-71,
teleks 815679

EO/757/89

COIG

CENTRALNY OŚRODEK INFORMATYKI GÓRNICTWA
40-065 KATOWICE UL. MIKOŁOWSKA 100
telefon – 574-777 telefaks – 517-442 teleks – 00313711

OFERUJE USŁUGI W ZAKRESIE:

– projektowania i oprogramowania systemów informatycznych;
– ujmowania i przetwarzania danych na komputerach MERA–9150, ODRA–1300, ICL 1900, 2900, 39, RIAD;
– kompletacji i sprzedaży systemów mikrokomputerowych, kopiarek i telekopiarek;
– instalacji, napraw i przeglądów okresowych urządzeń mikrokomputerowych, kopiarek i telekopiarek;
– szkolenia projektantów, programistów i operatorów sprzętu informatycznego.

OFERTY NASZE OBEJMUJĄ:

– udostępnienie eksploatowanych rozwiązań informatycznych oraz oprogramowania narzędziowego;
– projektowanie systemów informatyczych według założeń użytkowników lub na podstawie analizy jego potrzeb;
– ujmowanie danych źródłowych i ich przetwarzanie w trybie wsadowo-partiowym oraz zdalnym;
– udostępnianie czasu pracy maszyn cyfrowych;
– instalacje sieci lokalnych (Lan-Link, D-Link, Oa-Link);
– kompletacje i dostawę sprzętu mikrokomputerowego oraz kopiarek i telekopiarek;
– serwis techniczny gwarancyjny i pogwarancyjny sprzętu informatycznego realizowany w ciągu 48 godzin od wezwania klienta, dostosowanie drukarek D–180 do współpracy z IBM PC/XT/AT, przeróbki powiększające liczby znaków drukowanych do 158 w wierszu, rozszerzanie pamięci mikrokomputerów, wprowadzanie polskich liter do generatorów znaków;
– szkolenie informatyków w ośrodku wyposażonym w bogaty sprzęt dydaktyczny.

EO/943/89

Konferencja PTI

Współczesne obszary zastosowań informatyki

Śląski Oddział Polskiego Towarzystwa Informatycznego zorganizował w Szczyrku, w dniach 3–5 maja 1989 r. konferencję pn. „Współczesne obszary zastosowań informatyki". Intencją organizatorów konferencji było przedstawienie uczestnikom obszarów zastosowań informatyki, których istnienie być może nawet sobie uświadamiają, ale których na ogół nie próbują penetrować. Te braki w świadomości informatycznej profesjonalistów wynikają czasem z niedostępności środków, częściej jednak ze swego rodzaju gnuśności intelektualnej.

Wykłady wygłoszone w trakcie konferencji zorganizowano w taki sposób, że kilku wykładowców, na ogół z różnych środowisk, omawiało problem z różnych punktów widzenia.

Pierwszy cykl wykładów poświęcono „*Problemom sprzętowych uwarunkowań w konstrukcji systemu operacyjnego*". Wykładem o takim właśnie tytule rozpoczął konferencję dr Krzysztof Nałęcki. Po omówieniu przez prelegenta teoretycznych aspektów wpływu własności sprzętu na strukturę i parametry systemu operacyjnego, głos zabrał Adam Bukowy, który przedstawił „*Wykorzystanie własności sprzętu w systemie operacyjnym UNIX*". Te dwa wykłady uzupełnione zostały prelekcją na temat systemu ONX – konkretnej implementacji UNIX-a, wygłoszoną przez przedstawiciela firmy zajmującej się dystrybucją tego systemu w Polsce. Ku pewnemu zaskoczeniu organizatorów, po tych wystąpieniach wywiązała się dyskusja na tyle gorąca, że zaplanowana na pierwszy dzień część towarzyska spotkania uległa półgodzinnemu opóźnieniu.

Drugi dzień poświęcony był szeroko rozumianej problematyce technologii produkcji oprogramowania. Wykład dr. Janusza Zalewskiego zapoznał uczestników z obowiązującymi, głównie w USA, normami dotyczącymi zasad realizacji kilkunastu etapów okresu życia oprogramowania. Jakkolwiek można mieć wątpliwość (podzielaną przez część uczestników) co do celowości wprowadzania odpowiedników takich norm w Polsce, wobec mniejszej, niż w najbardziej rozwiniętych krajach, złożoności podejmowanych przedsięwzięć programistycznych, to jednak w dyskusji stwierdzono, że sformalizowanie procesu produkcji programów jest niezbędne, niezależnie od skali podejmowanych zadań. Wniosek taki wynika z faktu, że nawet w Polsce zaczyna powstawać przemysł oprogramowania, w którym metody, narzędzia i proces produkcji muszą odpowiadać zwyczajom i regułom wyznaczonym przez tę właśnie, przemysłową organizację produkcji. Z kolei mgr Andrzej Paprocki omówił warsztat pracy współczesnego programisty, składający się z coraz większej liczby narzędzi, o których stosowaniu, a nawet istnieniu, rodzimi profesjonaliści mają wciąż dość mgliste pojęcie. Udaną próbę stworzenia polskiego narzędzia tego rodzaju zaprezentował dr Zdzisław Szyjewski na przykładzie programu wykorzystującego propozycję standaryzacji procesu projektowania systemów informatycznych. Może przesadą byłoby stwierdzenie, że te trzy wykłady i wynikła po nich burzliwa dyskusja, otwarły nowe horyzonty przed zgromadzonymi informatykami i spowodowały przyjęcie nowych metod pracy, ale niewątpliwie dały one wszystkim impuls do zastanowienia się nad tym, jak w najbliższych latach powinien wyglądać zakład produkujący oprogramowanie. Niewątpliwie zaniedbania w zakresie korzystania z nowoczesnych technologii tworzenia produktów informatycznych są ogromne, ale z drugiej strony „*brak armat*" bardzo ogranicza możliwości wprowadzenia tych technologii.

Trzeci cykl wykładów, pozornie nie powiązanych tematycznie, łączyło hasło: „*co nowego w świecie i jak to wygląda u nas*". Inż. Piotr Fiszer ze swadą przedstawił obejmujący całą kulę ziemską system transmisji danych i funkcji aplikacyjnych SITA oraz zaprezentował zebranym stan zastosowań informatyki w PLL LOT. Wydało się to uczestnikom konferencji na tyle interesujące, że kilku z nich zaczęło się nawet zastanawiać nad zmianą pracy. Nie obeszło się oczywiście bez pytań na temat Boeinga. Z kolei doc. Wojciech Cholewa omawiający systemy eksportowe (doradcze) postawił tezę, że te zasady działania są bardzo proste, i co więcej, tezę skutecznie obronił, przedstawiając oprócz pojęć podstawowe udane krajowe realizacje takich systemów. Ostatni wykład tej serii, wygłoszony przez dr. Jerzego Solaka oraz mgr. inż. Janusza Rybnika dotyczył charakterystyki stacji roboczych (por. **INFORMATYKA**, nr 3, 4, 1989).

Poza wykładami planowymi odbyła się, na wzór szkół mrągowskich, sesja „*Oferta–problem*", w trakcie której reprezentanci pięciu firm przedstawili swoje propozycje. Problemów nikt nie zgłaszał.

Od strony organizacyjnej konferencja przebiegła w miarę sprawnie – dom wczasowy „Klimczok", w którym była zorganizowana, zapewnił więcej niż godziwe warunki pobytu; liczba nagłych zmian w programie kształtowała się na poziomie średniej krajowej. Poważnym niedopatrzeniem ze strony organizatorów była deszczowa i chłodna, jak na tę porę roku, pogoda.

dokończenie na III stronie okładki

ZAKŁAD KOMPUTEROWYCH SYSTEMÓW POMIAROWYCH
ul.Zb. z Bogdańca 4
80-419 Gdańsk
digimer
tel. 419519

** oferujemy profesjonalne, sprawdzone przez wielu użytkowników

programatory pamięci EPROM

z oprogramowaniem dla komputerów PC XT/AT:

● programator PCPE–512 umożliwia programowanie pamięci od 2716 do 27512(A) (CMOS), 2516/32/64, EEPROM 2816, 2864, 2916 oraz testowanie pamięci 8-bitowych SRAM od 2kB do 32kB

● Modułowe wyposażenie dodatkowe pozwala na:
– programowanie układów 8741/42/44/48/49/51/52
– symulację (E)PROM typ 2716/32/64/128/256/512 oraz 2 × 2716/32/64/128/256 8bit i 16bit

Udzielamy gwarancji na sprzęt i programy oraz zapewniamy serwis pogwarancyjny

Dostawa natychmiastowa. Niskie ceny oraz bonifikata przy zakupie 5 programatorów.

Nasze produkty spełnią Twoje oczekiwania

● Polecamy karty do komputerów PC XT/AT:
– tester scalonych układów cyfrowych TTL, CMOS i pamięci RAM (automatyczne rozpoznawanie typu układu scalonego, edytor wzorca testów)
– karta prototypowa XT (dekoder adresów i bufor)
– przetworniki A/C z separacją galwaniczną 500 V 12bit 8kan 10 V/20 mA 100 μs

● Przyjmujemy zamówienia na:
– system uruchomieniowy (in-circuit emulator) EMU51–8031/32/51/52
– bezobsługową centralkę telefoniczną MCT5211
– mikrofon bezprzewodowy 4101 (homologacja)

Prosimy o kontakt telefoniczny lub listowy:
– **udzielamy pełnej informacji**
– **uzgodnimy warunki dostawy**
– **wyślemy materiały informacyjne**

Jesteśmy zainteresowani współpracą w opracowaniu konstrukcji unikatowej aparatury pomiarowej i sterowników mikroprocesorowych.

EO/719/89

Infosystem '89

W kwietniu 1989 r. na terenach Targów Poznańskich odbyły się trzecie Międzynarodowe Targi Elektroniki, Telekomunikacji i Techniki Komputerowej *Infosystem*. Tym razem towarzyszyły im: Międzynarodowa Ekspozycja Maszyn Poligraficznych *Poligrafia* oraz XI Konferencja Naukowo-Techniczna *Polikon*. Pomysł połączenia wystawy komputerów i urządzeń poligraficznych jest na pewno rozsądny ze względu na przenikanie się obu dziedzin. Zdarzały się zresztą sytuacje, że nie było wiadomo, na której wystawie szukać danego wystawcy. Na przykład firma Rank Xerox, której oferta obejmuje głównie profesjonalny sprzęt małej poligrafii, miała stoisko w pawilonie poligraficznym, choć oferowała także koncesjonowane oprogramowanie sieciowe Netware (firmy Novell).

LICENCJONOWANE OPROGRAMOWANIE

W tym roku bodaj po raz pierwszy dużo mówiło się o handlu licencjonowanym oprogramowaniem renomowanych firm zachodnich oraz o ochronie oprogramowania przed piracким kopiowaniem. Łódzki *Aplikom* przedstawił swoją ofertę dotyczącą kompletnych, najnowszych wersji systemu projektowania inżynierskiego AutoCAD. Cena systemu nie jest mała – 12 mln zł, ale w zamian otrzymuje się pewność, że kupiony produkt jest tożsamy z oprogramowaniem rozprowadzanym w reszcie świata, a ponadto pełne wsparcie serwisowe (dwuletni serwis gwarancyjny), szkolenia, konsultacje itp. Producent przewidział sprzętowe zabezpieczenie swojego oprogramowania, tj. specjalną kostkę niezbędną do uruchomienia programu. W ten sposób użytkownik może wprawdzie mieć dowolnie wiele kopii, ale w tym samym czasie pracuje się zawsze tylko z jedną.

Obecnie oferuje się polską wersję językową AutoCAD-a. Podobno sprzedaż przebiega pomyślnie, a liczbę sprzedanych egzemplarzy można już liczyć w dziesiątkach sztuk, przy czym często jedna firma kupuje po kilka zestawów dla kilku komputerów. Przedstawiciel dostawcy wyraził opinię, że dopiero posiadanie legalnej, licencjonowanej kopii pozwala na praktyczne zastosowania AutoCAD-a do celów innych niż eksperymentalne.

Wrocławskie ZETO podjęło się dystrybucji oprogramowania firmy Microsoft. System operacyjny MS-DOS będzie rozprowadzany na podstawie licencji, wraz z komputerami Elwro 801 AT. Oznacza to prawo do rozpowszechniania zarówno programu, jak i dokumentacji. Podobno do podpisania kontraktu potrzebne było otrzymanie zgody na Bardzo Wysokim Szczeblu. Może następni chętni będą mieli ułatwioną drogę. W planach – przygotowanie spolszczonej wersji (do końca 1989 r.). W sprzedaży pozostałych programów ZETO będzie jedynie pośrednikiem. Przedstawiciel firmy Microsoft obecny na konferencji prasowej stwierdził, że w zasadzie system MS-DOS nie będzie rozprowadzany indywidualnie, lecz powinni go dostarczać producenci sprzętu, wykupując odpowiednią licencję. Każdy inny produkt programowy też może być dostarczony (istnieją wyjątki zawarte na liście embargowej COCOM: *LAN Manager* dla OS II i *SQL Server*). Odbiorca oprogramowania będzie podpisywał licencję, zobowiązując się do nierozpowszechniania go innym osobom. Wydaje się to być jedynym sposobem zapobieżenia nielegalnemu kopiowaniu w warunkach braku prawnej ochrony oprogramowania. Jeśli zapotrzebowanie na jakiś produkt okaże się wystarczająco duże, to może zostać przygotowana i wprowadzona do sprzedaży polska wersja językowa. Warto wspomnieć, że niektóre firmy już poprzednio legalnie spolszczyły programy, na które miały licencję, np. Rank Xerox od dwóch lat wyposaża swoje komputery w polskojęzyczny MS-DOS. Można ubolewać, że produkty tak spolszczone przez różne firmy nie są i nie będą identyczne. Teoretycznie ZETO nie jest wyłącznym dystrybutorem, lecz w praktyce firma Microsoft stara się nie dopuszczać do konkurencji między swoimi przedstawicielstwami.

SPRZĘT DLA KOMPUTERÓW OSOBISTYCH

Firma *Atomica* współpracująca z Polską Akademią Nauk, oferowała stacje pamięci na dyskach optycznych (CD-ROM). Nie są one raczej przeznaczone dla indywidualnych amatorów – cena podstawowego zestawu wynosi co najmniej 15 mln zł, a późniejsza prenumerata jednej bazy danych na dysku o pojemności 600 MB (z ciągłymi poprawkami) – ok. 50 mln. Firma liczy na ośrodki dokumentacji naukowo-technicznej i inne tego rodzaju instytucje. Firma *Data Start* prezentowała (chyba po raz pierwszy w Polsce) mikrokomputer PS-2/80.

Postęp dotyczy także przenośnych komputerów podręcznych (ang. *laptop*). Firma *Walkom* prezentowała model LP-3300, z procesorem 286, stanowiący odpowiednik AT. Ma on bardzo dobry i czytelny ekran ciekłokrystaliczny (720 × 400 punktów), ale można także dołączyć zewnętrzny monitor klasy EGA. Komputer ma wbudowany dysk stały o pojemności 40 MB i stację dyskietek 3,5-calowych (1,44 MB), można też dodać zewnętrzną stację dyskietek o średnicy 5,25 cala. Łącze szeregowe pozwala dołączyć modem, a równoległe – drukarkę. Można także dołożyć inne karty komputera PC. Cena – ok. 3100 dolarów. Podobną cenę i możliwości miał komputer IMC 9000, także z procesorem 286, doskonałym ekranem plazmowym o rozdzielczości 640 × 400 punktów i kartą EGA, stacją dyskietek 3,5-calowych i dyskiem stałym o pojemności 20 MB.

Spośród drukarek szczególne wrażenie robiły kolorowe drukarki strumieniowe (ang. *jet printer*) firmy *Seikosha*, oferowane przez firmę *Soft-Tronik*. Zachwyt nawet u najbardziej wybrednych mogła budzić jakość wydruku, czystość kolorów i liczba odcieni uzyskanych z podstawowych trzech barw – żółtej, niebieskiej i czerwonej – oraz oczywiście czarnej. Niestety, drukarki te wymagają specjalnego papieru i tuszu, co jest zrozumiałe, ale w naszych warunkach uciążliwe.

Również firma *Epson* była reprezentowana przez swoje przedstawicielstwo IMC. Prezentowano całą linię drukarek, od tradycyjnych dziewięcioigłowych do laserowych (wsad drukujący firmy *Ricoh*). Odbyła się także konferencja prasowa przybyłych przedstawicieli Epsona w Europie. Na pytanie o sposoby osiągnięcia tak wysokiej jakości produkowanego sprzętu odpowiedziano, że Epson wywodzi się z firmy Seiko produkującej zegarki. Przyznano jednak, że firma nie stała się potentatem na rynku drukarek laserowych, lecz planuje zajęcie dobrego miejsca wśród producentów drukarek strumieniowych. Obecnie jest oferowana drukarka TSQ-4800, z 48 dyszami sterowanymi pompami piezoelektrycznymi, drukująca do 600 znaków/s w trybie zwykłym i 200 znaków/s w trybie korespondencyjnym. Można oczywiście drukować także obrazy graficzne z maksymalną rozdzielczością 360 punktów/cal. Drukarka akceptuje różne rodzaje papieru, także składankę, co jest dużą zaletą w porównaniu z drukarkami laserowymi.

MINIKOMPUTERY

Choć większość ekspozycji stanowiły mikrokomputery osobiste, to jednak był też poważniejszy sprzęt. Na stanowisku firmy *ComTech* widniał nawet napis: *To nie jest IBM!* I rzeczywiście, prezentowano kopię PDP-11, z radzieckim procesorem i zachodnimi urządzeniami. Firma *Amepol* natomiast wystawiła system komputerowy MX-16, korzystający z modułu procesora Mery 400.
Okazuje się, że polskie procesory wcale nie są bezużyteczne, oczywiście pod warunkiem wyposażenia ich w solidne urządzenia na poziomie światowym. System korzysta z oryginalnego oprogramowania, opartego na systemie operacyjnym CROOK-5. Ośmiostanowiskowy system do przetwarzania danych kosztuje ok. 50 mln zł. Planuje się następne wersje, zapewne już korzystające z mikroprocesorów serii Motorola 68000. Jak wynika z opisu, również systemy DP 2000 firmy *Danpol* oparto na procesorach Motoroli. Są to rozbudowane, wielostanowiskowe zestawy, mające dalsze możliwości rozbudowy (do 120 MB pamięci operacyjnej, 4 GB pamięci dyskowej, 600 terminali). System korzysta z Unixa 5.3 i oprogramowania z nim związanego (np. systemu zarządzania bazą danych *Informix*). Nowym produktem firmy jest koncentrator komunikacyjny DP 6300 MDCC, pozwalający dołączyć do 64 terminali przez jedno łącze kablowe. A firma *Gambit* nadal rewitalizuje Mery 9150 (odpowiedniki *Seechecka*), co pozwala mieć prawie nowy wielostanowiskowy system wprowadzania danych za kilkanaście milionów, zamiast całkiem nowego za 200 mln.

Coraz więcej państw wschodniej Europy

zaczyna produkować odpowiedniki superminikomputerów serii VAX. Do Jugosławii dołączyły Czechosłowacja i NRD. Czechosłowacki VAX 11/780 nazywa się SM 52/12 i jest wyposażony w stacje pamięci taśmowej o gęstości zapisu 800/1600 bitów na cal, dyski wymienne o pojemności 200 MB, dyski Winchester – 160 MB i stacje graficzne Tesla. Niestety urządzenia nie robią dobrego wrażenia, a doświadczenia z mniejszymi minikomputerami serii SM pozwalają spodziewać się najgorszego. To samo dotyczy niemieckiego Robotrona 1840. Nie było żadnej reprezentacji ZSRR.

Warszawski Ośrodek Postępu Technicznego oferował nawet dostawę używanych komputerów IBM serii 43xx – pośredni szczebel między dużymi minikomputerami a komputerami głównego szeregu (ang. *mainframe*). Elwro natomiast pokazało działający dwumaszynowy zestaw R-34 oraz (nowość) modem M-2424 działający z szybkością 2400 bitów/s.

Nie zabrakło tradycyjnych wystawców. Firma *ZOT* PAN przedstawiała sprzęt *Hewlett-Packarda*. Oprócz komputerów serii 3000 (z nowym – najmniejszym 3000 LX), także nową dwustronną drukarkę LaserJet IID i nowe pamięci dyskowe Winchester. Pojedyncza stacja (6 dysków o średnicy 5,25 cala) ma pojemność 304 MB i czas dostępu 17 ms, a zestaw z zasilaczem może obejmować trzy takie stacje. Niepokoił natomiast brak (chyba po raz pierwszy od wielu lat) stoiska ICL.

STANOWISKA INNYCH WYSTAWCÓW

Nowością był cały ciąg stanowisk zorganizowanych przez *Fundację Edukacji Komputerowej*. Każde stanowisko tworzyło niejako jedną klasę, wyposażoną przez jednego dostawcę. W polskiej klasie stały ośmiobitowe komputery Bosman B, produkcji Unimoru. Były też klasy wyposażone przez Czechosłowację, Norwegię, Bułgarię, Wielką Brytanię i Austrię.

Na wystawie poligraficznej również prezentowano wiele zastosowań komputerów. Już tradycyjnie pokazywano zestawy Poltype do komputerowego składu tekstów o różnym stopniu złożoności, zwłaszcza tekstów matematycznych. Pojawiła się nowa generacja tych urządzeń – modele 05 i 06 – z najnowszymi rozszerzeniami oprogramowania (m.in. dzielenie wyrazów dla 12 języków, wprowadzanie obrazów za pomocą skanera) oraz atrakcyjną możliwością dołączenia 20-calowego monitora o zwiększonej rozdzielczości, umożliwiającego podgląd typograficzny pełnej strony formatu A3. Pokazywano także wielostanowiskowe zestawy do składu komputerowego. Niestety, większość z nich jest dostosowana do pojedynczych fotonaświetlarek. Właściwie jedyną metodą przeniesienia na fotonaświetlarkę dokumentu przygotowanego przez program uniwersalny, taki jak np. *Ventura*, jest zapisanie go w postaci programu w języku *Postscript*, ale taki program nie jest z kolei akceptowany przez każdą fotonaświetlarkę. Tzw. *Raster Image Processor*, interpretujący programy w Postscripcie i sterujący naświetlarkami, kosztuje przeszło 50 tys. marek RFN!

Tradycyjnie już przy targach funkcjonowało stoisko Domu Książki, tym razem niemal całkowicie zdominowane przez WNT (obchodzące w tym roku swoje czterdziestolecie). Praktycznie przez cały czas stoisko było oblegane przez kupujących, książki „szły jak woda" i to nie tylko te najnowsze. Potwierdza to jedynie znaną prawdę, że dystrybucja jest u nas fatalna, wskutek czego w różnych częściach kraju brakuje poszukiwanych książek, gdy tymczasem w magazynach jest ich pod dostatkiem.

Przy okazji kupowania książek, spostrzegawczy mogli zapewne zauważyć ulotkę informującą o powstaniu Agencji Obrotu Oprogramowaniem, która ma zamiar zajmować się m.in. ochroną prawną oprogramowania. Niewiele więcej można było dowiedzieć się o tej inicjatywie (nie pojawił się bowiem nigdzie przedstawiciel Agencji), ale z pewnością należy uznać za cenną każdą próbę „unormalnienia" sytuacji oprogramowania na polskim rynku. Jedni zatrą ręce z radości, inni z pewnością zadrżą ze strachu – tak czy inaczej na efekty działania Agencji trzeba będzie jeszcze jakiś czas poczekać.

JAROSŁAW DEMINET
HANNA WŁODARSKA

digitex

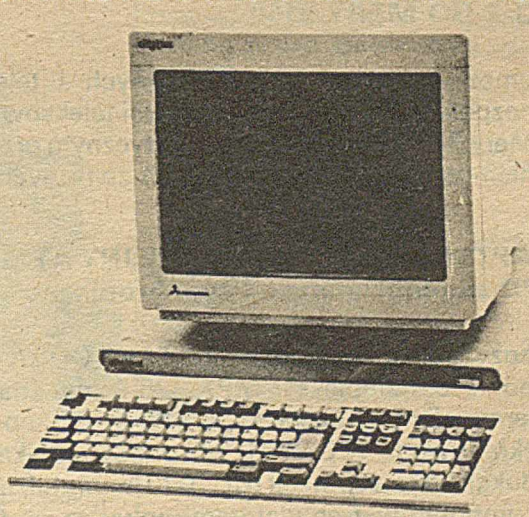
ZAKŁAD SYSTEMÓW CYFROWYCH
81-832 Sopot, ul. Mickiewicza 20
tel. 51-28-27, tlx. 512290 dgtex pl

KOLOROWY TERMINAL GRAFICZNY DXT–240

- emuluje protokół terminali graficznych **DEC-a VT–240, VT–125** i **TEKTRONIX 4014**
- emuluje protokół terminali alfanumerycznych DEC-a VT–220, VT–100, VT–52
- przeznaczony do pracy w konfiguracjach wielodostępnych z komputerami serii VAX, SM, PDP11, PC XT/AT pod kontrolą systemów operacyjnych XENIX, UNIX, RSX, MULTILINK i innych
- wyświetla informacje nie tylko w postaci znaków alfanumerycznych, ale również w postaci wykresów, rysunków itp., zgodnie z protokołem graficznym **ReGIS**, który m.in. zapewnia kreślenie prostych, okręgów, elips, krzywych, zapełnianie, „dostęp" do każdego punktu itp.
- rozdzielczość grafiki:
  640 × 256 punktów – DXT–240A
  640 × 400 " – DXT–240B
  640 × 480 " – DXT–240C
  800 × 480 " – DXT–240D
- atrybuty: 16 kolorów każdego punktu (lub 8 kolorów i migotanie)
- tryb alfanumeryczny: max. 60 wierszy po 132 znaki
- interface komunikacyjny RS 232 lub pętla prądowa
- interface drukarki Centronix lub RS 232
- klawiatura typu PC–XT
- monitor kolorowy 14" (RGB standard, EGA, Multisync – zależnie od rozdzielczości terminala) lub monitor monochromatyczny.

☆ DEC, VT, ReGIS są zastrzeżonymi znakami firmowymi Digital Equipment Co.
☆ TEKTRONIX jest zastrzeżonym znakiem firmy Tektronix Inc.

EO/646/89



SWEDEX UNIVERSAL Co.

PRZEDSIĘBIORSTWO ZAGRANICZNE W POLSCE

Biuro: ul. Kolejowa 2, 05–250 Radzymin
Teleks 815888 swedx pl, telefon: Warszawa 762004 w. 13 lub 365
Zakład Elektroniki: ul. Wspólna 1, 05–440 Wesoła-Zielona
Teleks 813935 swedx pl, telefon: Warszawa 153365

OFERTA ZAKŁADU ELEKTRONIKI

MODEM TRANSMISJI DANYCH 1230P

- umożliwia transmisję dwukierunkową jednoczesną (full duplex) po łączach telefonicznych komutowanych i dzierżawionych: w trybie asynchronicznym z szybkościami 0–300 b/s, 600 b/s i 1200 b/s; w trybie synchronicznym z szybkością 1200 b/s;
- zgodny z zaleceniami CCITT V.21, V.22;
- posiada świadectwo homologacji Instytutu Łączności PRL.

KONCENTRATOR STANOWISK OPERATORSKICH KSO-4 DO REJESTRATORA DANYCH MERA 9150 (SEECHEK)

- pozwala na dołączenie do Mery 9150 poprzez jedno łącze transmisji danych do 4 stanowisk operatorskich i drukarki systemowej;
- nie wymaga zmian w systemie operacyjnym i sprzęcie rejestratora;
- posiada wbudowane testy pozwalające na szybką lokalizację uszkodzeń i sprawdzenie łącza transmisji danych;
- jest produkowany także w wersji z wbudowanym modemem stałoprądowym KN-9600.

ADAPTER TELEGRAFICZNY ATG-2 DO MERY 9150

- umożliwia wprowadzanie danych i teleksów bezpośrednio z dalekopisu (sieci teleksowej) do Mery 9150 z kontrolą i automatycznym przesyłaniem komunikatów (teleksów) zwrotnych.

ADAPTER TELEKOMUNIKACYJNY AT 8-MPX (UPD asynchroniczne i synchroniczne)

- pozwala na dołączenie do EMC ODRA 1305 poprzez multiplekser MPX 325 ośmiu zdalnych terminali np. drukarki DZM 180 KSRE, monitora ekranowego ICL 7181 lub odpowiednika produkcji krajowej, stacji ICL 7020, mini- lub mikrokomputerów z emulatorami powyższych terminali;
- realizuje automatyczne rozłączenie połączeń komutowanych.

INTERFEJSY POMIAROWE SU-GPIB I SU-GPIBF STANOWIĄCE PEŁNĄ IMPLEMENTACJĘ NORMY IEC-625 (IEEE 488) UMOŻLIWIAJĄCE STEROWANIE PROCESEM POMIAROWYM Z SZYBKOŚCIĄ DO 500 kB/s

- oprogramowanie: procedury wywoływane z takich języków programowania, jak C, PASCAL, FORTRAN, BASIC;
- układ interfejsu zrealizowany jest w oparciu o kontroler 7210 oraz bufory firmy Texas Instruments.

PAKIET TRANSMISJI SZEREGOWEJ SYNCHRONICZNEJ/ASYNCHRONICZNEJ BSC DO KOMPUTERÓW KOMPATYBILNYCH Z IBM PC XT/AT

- umożliwia pracę komputera jako terminala np. ICL 7181, ICL 7020, IBM 2780, IBM 3780, IBM 3270;
- dostarczamy oprogramowanie emulacyjne pakietu BSC.

PROGRAMATOR PAMIĘCI EPROM ORAZ MIKROKOMPUTERÓW JEDNOUKŁADOWYCH

- umożliwia programowanie pamięci typu 2716-27512 i ich funkcjonalnych odpowiedników o napięciach programujących 12,5; 21; 25 V oraz mikrokomputerów jednoukładowych 8741, 8748, 8749, 8751, 8755;
- automatycznie wybiera parametry programowania;
- współpracuje z komputerem przez styk RS-232 (V.24);
- oprogramowanie do komputerów IBM PC XT/AT.

KOMPUTERY SU 88-PC W PEŁNI KOMPATYBILNE Z IBM PC KONFIGUROWANE ZGODNIE Z ŻYCZENIEM KLIENTA.

EO/1263/88

Bezpieczeństwo zapisu danych – Produkcja firmy BASF

W 1934 roku firma BASF dokonała przełomu technologicznego uzyskując pierwszą na skalę przemysłową produkcję magnetycznego nośnika zapisu informacji. Na tej samej zasadzie, zapisu magnetycznego – pierwotnie stosowanego do zapisu dźwięku – opiera się jeszcze dzisiaj zapis dźwięku, obrazu i danych. Stały postęp i produkcja najwyższej jakości nośników magnetycznych wymaga ogromnej wiedzy w zakresie chemii i znajomości "Know-how".

W wielu dziedzinach firma BASF osiągnęła wiodącą w skali światowej pozycję poprzez wieloletnie badania i wprowadzanie nowych technologii.

Od wielu lat firma BASF jest kompetentnym partnerem handlowym Polski. Firma BASF posiada liczne, technologicznie dojrzałe rozwiązania problemów w dziedzinach chemikalii, barwników, środków pomocniczych, tworzyw, surowców i energii, produktów dla rolnictwa i wyrobów powszechnego użytku.

Produkty firmy BASF torują drogę w nowoczesnej chemii. Ważnym czynnikiem powodzenia jest owocna współpraca specjalistów nauk przyrodniczych i dyscyplin technicznych, od klasycznej chemii i techniki, przez fizykę, aż do biologii i medycyny. W 1987 roku wzrosły w firmie BASF nakłady na badania i rozwój ponownie o 6%, t. j. do 1.613 mln marek.

Punkty ciężkości rozwoju skierowane są na produkty powszechnego użytku, do których należą też nośniki magnetyczne.

Bezpieczeństwo i niezawodność

Szczególnie w przechowywaniu danych ważnym jest ich długotrwałe zachowanie. Zabezpieczenie na dłuższy czas cennych zbiorów danych jest bardzo ważnym wymogiem dla nauki i gospodarki.
Nośniki pamięci firmy BASF wykazują tu najwyższą jakość. Olbrzymie nakłady na badania i rozwój oraz własne nowoczesne miejsca produkcji zapewniają firmie pierwszeństwo w technologii zapisu danych.

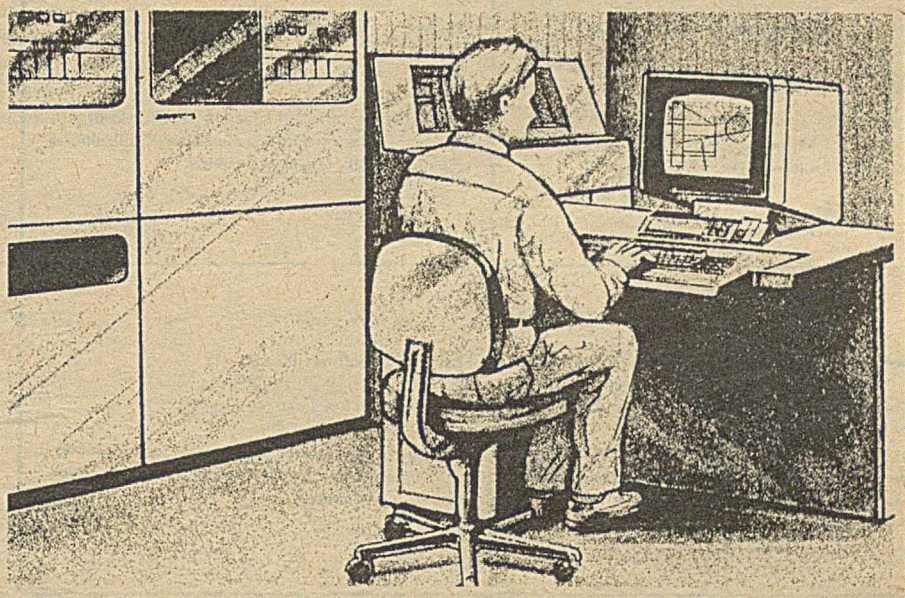
Od nośników danych, po systemy przetwarzania

Kompletne systemy przetwarzania danych dopełniają ofertę firmy BASF w dziedzinie przetwarzania danych.

Już w 1970 roku firma BASF była jedną z pierwszych, która wyszła na rynek z urządzeniami peryferyjnymi, kompatybilnymi co do łącza (tzw. PCM).
Od 1980 roku rozszerzono program na kompletne systemy produkowane w Japonii. Na rynku polskim proponujemy:

- Jednostkę centralną BASF 7/6X zajmującą powierzchnię tylko 1 m2 – najmniejszą jednostkę tej klasy z możliwością rozbudowy, spełnia wszystkie indywidualne wymagania użytkownika.
- Pamięci taśmowe, małe o zwartej budowie, dla których kontroler (sterownik) zintegrowany jest w pierwszym przewijaku łańcucha z możliwością pracy w trzech gęstościach zapisu (800/1600/6250 bpi).
- Drukarki wierszowe o dużej prędkości, małych wymiarach, zwartej budowie i wyjątkowo ciche. Prędkości od 1500 do 2000 linii/min. Osiągane w tej samej obudowie, modułowo rozbudowywalnej z modelu do modelu, z nadzwyczajną jakością druku, przy zastosowaniu taśmy stalowej z trzcionkami. Obecnie około 200 taśm z różnym rodzajem pisma.

Ponadto:
- System telekomunikacji lokalnej i zdalnej.



- Napędy dyskowe ze stałą głowicą (tzw. HDA) o pojemności do 317,5 MB na jednostkę.

Nasz system oparty na nowoczesnej technologii pozwala na oszczędność miejsca i małe zużycie prądu, jest odporny na wahania częstotliwości i napięcia zasilania. Doświadczenie firmy BASF i wypróbowana technika gwarantują niezawodną pracę systemu.

Uwaga!
Sprzęt firmy BASF jest kompatybilny nie tylko ze sprzętem IBM, ale także RIAD.

Przedstawicielstwo BASF w Polsce:

LIM Centre
Al. Jerozolimskie 65/79
00–697 Warszawa

telefony: 300291-5
teleks: 816877 lub 816878
telefax: 300296

EO/1211/89

Elastyczne nośniki danych jak dyskietki, taśmy komputerowe, kasety i kasety Cartridge oraz kasety i pakiety dyskowe będące w programie sprzedaży firmy BASF.

Ich niezawodna funkcjonalność działania podwyższa niezawodność całego systemu rejestracji danych.

BASF

Hołyński M.: Sprzęg graficzny jako system ekspertowy

INFORMATYKA 1990, nr 1, s. 2

Charakterystyka najnowszych osiągnięć oraz perspektywy rozwoju w dziedzinie systemów ekspertowych z zastosowaniem grafiki komputerowej.

Hołyński M.: Graphics interface as an expert system

INFORMATYKA 1990, No. 1, p. 2

Characteristics of the newest achievements and development prospects of expert systems with graphic representation.

Hołyński M.: Graphische Schnittstelle als ein Expertensystem

INFORMATYKA 1990, Nr. 1, S. 2

Eine Charakteristik der neuesten Errungenschaften und Entwicklungsaussichten im Bereich der Expertensysteme mit Anwendung der Computergraphik.

Dydek A.: Porządkowanie zbiorów danych według polskiego alfabetu

INFORMATYKA 1990, nr 1, s. 6

Propozycje praktycznych rozwiązań porządkowania zbiorów danych według polskiego alfabetu oraz analiza wyników badań efektywności różnych wariantów porządkowania zbioru.

Dydek A.: Data file sequencing according to polish alphabet

INFORMATYKA 1990, No. 1, p. 6

Proposals of practical solutions for data file sequencing according to polish alphabet and analysis of results from effectiveness research of file seqencing different variants.

Dydek A.: Dateiordnen nach polnischen Alphabet

INFORMATYKA 1990, Nr. 1, S. 6

Ein Vorschlag für praktische Lösungen der Dateiordnens nach polnischen Alphabet und die Ergebnisse der Effektivitätsanalyse von verschiedenen Varianten des Dateiordnens.

Paprocki A.: Problemy implementacji języka Ada

INFORMATYKA 1990, nr 1, s. 9

Charakteryzacja aktualnej sytuacji w dziedzinie implementacji języka Ada oraz omówienie trudności w realizacji kompilatorów tego języka.

Paprocki A.: Ada implementation problems

INFORMATYKA 1990, No. 1, p. 9

Characteristics of actual situation in the field of Ada language implementation and discussion of difficulties in realization of the language compilers.

Paprocki A.: Probleme der Ada–Implementierung

INFORMATYKA 1990, Nr. 1, S. 9

Eine Charakteristik der aktuellen Lage im Bereich der Ada–Implementierung und eine Besprechung der Schwierigkeiten in Realisierung von Kompilatoren dieser Sprache.

Zalewski J.: Okres istnienia oprogramowania (1)

INFORMATYKA 1990, nr 1, s. 12

Pierwsza część omówienia norm amerykańskich dotyczących inżynierii oprogramowania oraz organizacji procesu wytwarzania produktów programowych.

Zalewski J.: Software life cycle (1)

INFORMATYKA 1990, No. 1, p. 12

First part of discussion of american standards for software engineering and organization of program products manufacturing process.

Zalewski J.: Lebenszyklus der Software (1)

INFORMATYKA 1990, Nr. 1, S. 12

Erster Teil einer Besprechung von amerikanischen Normen für Softwaretechnologie und Organisation des Programmprodukteerzeugungsprozesses.

Huzar Z., Kuźniarz L.: Język Lotos (1). Specyfikacja typów danych

INFORMATYKA 1990, nr 1, s. 19

Pierwsza część charakterystyki języka Lotos jako narzędzia specyfikacji formalnej, zawierająca wprowadzenie do specyfikacji abstrakcyjnych typów danych.

Huzar Z., Kuźniarz L.: The Lotos language (1). Data type specification

INFORMATYKA 1990, No. 1, p. 19

First part of the Lotos language characteristics as a tool for formal specification, which includes the introduction for abstract data type specification.

Huzar Z., Kuźniarz L.: Die Lotos–Sprache (1). Datentypspezifikation

INFORMATYKA 1990, Nr. 1, S. 19

Erster Teil einer Charakteristik der Lotos–Sprache als ein Werkzeug für Formalspezifikation, die eine Einführung zur Spezifizierung der abstrakten Datentypen umfasst.

Gondzio M., Pawłowski M., Wytrębowicz J.: Uruchamianie urządzeń mikroprogramowanych (2)

INFORMATYKA 1990, nr 1, s. 24

Druga część charakterystyki problemu wspomagania procesu uruchamiania urządzeń mikroprogramowanych, zawierająca omówienie oprogramowania systemu MIDES, zrealizowanego i stosowanego do takich zadań w Instytucie Informatyki Politechniki Warszawskiej.

Gondzio M., Pawłowski M., Wytrębowicz J.: Developing of microprogrammed devices (2)

INFORMATYKA 1990, No. 1, p. 24

Second part of characteristics of computer assisted development process of microprogrammed devices, which contains discussion of software of the MIDES system, elaborated and applied for such tasks in the Informatics Institute of Warsaw Technical University.

Gondzio M., Pawłowski M., Wytrębowicz J.: Inbetriebsetzung der mikroprogrammierten Einrichtungen (2)

INFORMATYKA 1990, Nr. 1, S. 24

Zweiter Teil einer Charakteristik der Unterstützung von Inbetriebsetzungsprozess der mikroprogrammierten Einrichtungen, der eine Besprechung von Software des im Institut für Informatik der Warschauen Technischen Universität realisierten und angewendenten MIDES-Systems umfasst.

Polskie znaki i klawiatury w normach międzynarodowych

Artykuł o porządkowaniu zbiorów według polskiego alfabetu (str. 6) sprowokował mnie do odszukania kilku norm międzynarodowych, traktujących o komputerowej reprezentacji znaków narodowych. Na poziomie międzynarodowym, najwcześniej zaczęło zajmować się tą sprawą Europejskie Stowarzyszenie Producentów Komputerów, ECMA (ang. *European Computer Manufacturers Association*), z siedzibą w Genewie, opracowując kilka dokumentów znanych jako podstawa normy ECMA [1–3]. One właśnie posłużyły później jako podstawa do opracowania odpowiednich norm ISO.

Pierwszą normą ISO obejmującą reprezentację znaków polskiego alfabetu była norma ISO 6937/2 [4]. Polskie litery są w niej reprezentowane jako ciągi dwubajtowe, odpowiadające złożeniu litery oryginalnego alfabetu łacińskiego i właściwego znaku diakrytycznego. Ciągi kodowe dla tych liter podano w tabeli 1. W operowaniu tą metodą polskimi znakami jest jednak konieczne użycie klawisza cofnięcia (ang. *backspace*), co może zasadniczo skomplikować ich automatyczne przetwarzanie. Nowszą normą ISO 8859/2 [5] objęto więc reprezentowanie znaków narodowych za pomocą pojedynczych bajtów, umieszczając w niej także znaki polskiego alfabetu (tabela 2). Jest to tzw. kod ISO Latin 2.

Istnienie wymienionych norm nie załatwia jednak wszystkiego, choć znacznie ułatwia przetwarzanie i wyprowadzanie polskich liter. Nie mniej ważne, a może jeszcze ważniejsze jest bowiem ich wprowadzanie, z czym wiąże się rozmieszczenie odpowiednich klawiszy na klawiaturze. Tego aspektu dotyczy dokument ECMA-115 [3], określający rozmieszczenie klawiszy odpowiadających znakom diakrytycznym – dla sytuacji objętych normą ISO 6937/2, a więc odnoszących się do dwubajtowej reprezentacji polskich znaków. Układ klawiszy dla jednobajtowej, a więc wygodniejszej, reprezentacji znaków narodowych zdefiniowano w projekcie normy ISO DIS 8884, posługując się dodatkowym klawiszem *SHIFT*.

JANUSZ ZALEWSKI

LITERATURA

[1] ECMA: 8-bit Single-Byte Coded Graphic Character Set. Latin Alphabets No. 1 to No. 4. Standard ECMA-94, European Computer Manufacturers Association, Geneva, June 1986

[2] ECMA: 8-bit Single-Byte Coded Graphic Character Sets. Latin/Cyryllic Alphabet. Standard ECMA-113, European Computer Manufacturers Association, Geneva, June 1986

[3] ECMA: Common Secondary Keyboard Layout for Languages Using a Latin Alphabet. Standard ECMA-115, European Computer Manufacturers Association, Geneva, June 1986

[4] ISO: Information Processing – Coded Character Sets for Text Communication. Part 2 – Latin Alphabetic and Non-alphabetic Graphic Characters. ISO 6937/2. International Organization for Standardization, Geneva, December 1983

[5] ISO: Information Processing – 8-bit Single-Byte Coded Graphic Character Sets. Part 2 – Latin Alphabet No. 2. ISO 8859-2, International Organization for Standardization, Geneva, February 1987.

Tabela 1. Ciągi kodowe odpowiadające polskim literom według normy ISO 6937/2, w kolejności: litera alfabetu łacińskiego, znak diakrytyczny (litery ł i Ł mają ciągi jednobajtowe)

Litera	ą	Ą	ć	Ć	ę	Ę	ł	Ł	ń	Ń	ó	Ó	ś	Ś	ź	Ź	ż	Ż
Ciąg dziesiętny	97/206	65/206	99/194	67/194	101/206	69/206	248	232	110/194	78/194	111/194	79/194	115/194	83/194	122/194	90/194	122/199	90/199
Ciąg szesnastkowy	61CEH	41CEH	63C2H	43C2H	65CEH	45CEH	F8	E8	6EC2H	4EC2H	6FC2H	4FC2H	73C2H	53C2H	7AC2H	5AC2H	7AC7H	5AC7H

Tabela 2. Ciągi kodowe odpowiadające polskim literom według normy ISO 8859-2

Litera	ą	Ą	ć	Ć	ę	Ę	ł	Ł	ń	Ń	ó	Ó	ś	Ś	ź	Ź	ż	Ż
Ciąg dziesiętny	177	161	230	198	234	202	179	163	241	209	243	211	182	166	188	172	191	175
Ciąg szesnastkowy	B1H	A1H	E6H	C6H	EAH	CAH	B3H	A3H	F1H	D1H	F3H	D3H	B6H	A6H	BCH	ACH	BFH	AFH

Specyfikacja typów danych

dokończenie ze s. 22

Wykaz typów standardowych jest podany w [1]. Należą do nich między innymi typy o sygnaturach podanych na wydruku 7, przy czym *NonEmptyString* jest typem sparametryzowanym, definiującym ciągi złożone z elementów należących do zbioru formalnego *Element*.

LITERATURA

[1] Brinksma E. (Ed.): Information processing systems. Open systems interconnection. LOTOS – a formal description technique based on temporal ordering of observational behaviour. ISO 8807, 1988

[2] Ehrig H., Mahr B.: Fundamentals of algebraic specification. Part I: Equations and initials semantics. Springer-Verlag, Berlin, 1985

[3] Goguen J. A.: One, pope, a hundred thousand specification languages. 1986

[4] Huzar Z., Kuźniarz L.: LOTOS – język specyfikacji rozproszonych systemów komputerowych. Centrum Obliczeniowe Politechniki Wrocławskiej, Raport SPR nr 25/88, 1988

[5] ISO/TC97/SC16. Reference model for Open Systems Interconnections. ISO 7998, 1984

[6] Milner R.: A calculus of communicating systems. Lecture Notes in Computer Science, Vol. 92, Springer-Verlag, Berlin, 1980.

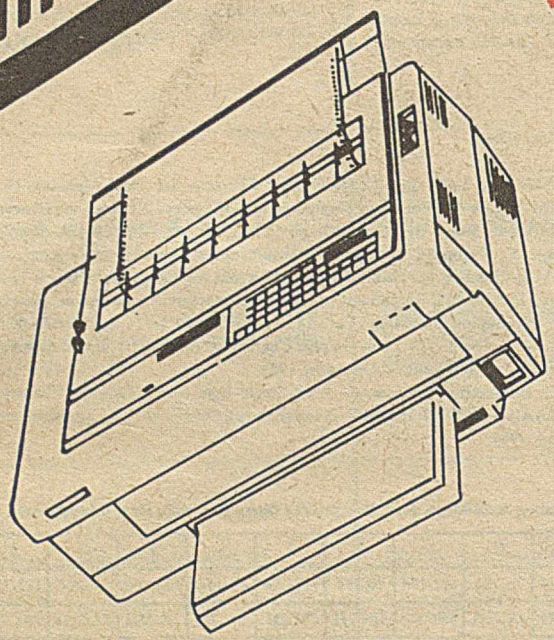
Współczesne obszary zastosowań informatyki

dokończenie ze s. 27

Materiały uczestnicy otrzymali w postaci teczki z luźnymi odbitkami referatów, jednak Śląski Oddział Polskiego Towarzystwa Informatycznego wyda w swojej serii opracowań monograficznych zeszyty poświęcone omawianej tematyce, które uczestnicy seminarium i wszyscy członkowie PTI zrzeszeni w Śląskim Oddziale będą sukcesywnie otrzymywać. Niedociągnięcia pierwszej kilkudniowej konferencji (jednodniowe spotkania, głównie z producentami sprzętu informatycznego, prowadzi Oddział od czasu swego powstania) organizatorzy zobowiązali się poprawić na kolejnej konferencji, która powinna odbyć się wiosną 1990.

PIOTR FUGLEWICZ

SEIKOSHA
SEIKOSHA
SEIKOSHA
SEIKOSHA
SEIKOSHA
SEIKOSHA
SEIKOSHA
SEIKOSHA
SEIKOSHA



Japoński koncern SEIKOSHA,
produkujący znane zegarki SEIKO,
migawki do fotoaparatów,
drukarki komputerowe,
zbliżył się do naszych rynków, otwierając
filię w Hamburgu,
produkującą pełną gamę drukarek komputerowych.

Informacje techniczne, katalogi:
SOFT-TRONIK SERVICE
00-710 Warszawa
ul. Idzikowskiego 2
tel. 40-46-79, fax (02) 635-21-95
telex 816075

EO/717/89